

Efficient Reassembling of Graphs, Part 1: The Linear Case

Assaf Kfoury*
Boston University

Saber Mirzaei*
Boston University

Abstract

The *reassembling of a simple connected graph* $G = (V, E)$ is an abstraction of a problem arising in earlier studies of network analysis. Its simplest formulation is in two steps:

- (1) We cut every edge of G into two halves, thus obtaining a collection of $n = |V|$ one-vertex components, such that for every $v \in V$ the one-vertex component $\{v\}$ has $|degree(v)|$ half edges attached to it.
- (2) We splice the two halves of every edge together, not of all the edges at once, but in some ordering Θ of the edges that minimizes two measures that depend on the edge-boundary degrees of assembled components.

A component A is a subset of V and its edge-boundary degree is the number of edges in G with one endpoint in A and one endpoint in $V - A$ (which is the same as the number of half edges attached to A after all edges with both endpoints in A have been spliced together). The **maximum** edge-boundary degree encountered during the reassembling process is what we call the **α -measure** of the reassembling, and the **sum** of all edge-boundary degrees is its **β -measure**. The α -optimization (resp. β -optimization) of the reassembling of G is to determine an order Θ for splicing the edges that minimizes its α -measure (resp. β -measure).

There are different forms of reassembling, depending on restrictions and variations on the ordering Θ of the edges. We consider only cases satisfying the condition that if the an edge between disjoint components A and B is spliced, then all the edges between A and B are spliced at the same time. In this report, we examine the particular case of *linear reassembling*, which requires that the next edge to be spliced must be adjacent to an already spliced edge. We delay other forms of reassembling to follow-up reports.

We prove that α -optimization of linear reassembling and *minimum-cutwidth linear arrangement* (CutWidth) are polynomially reducible to each other, and that β -optimization of linear reassembling and *minimum-cost linear arrangement* (MinArr) are polynomially reducible to each other. The known NP-hardness of CutWidth and MinArr imply the NP-hardness of α -optimization and β -optimization.

1 Introduction

We start with a gentle presentation of our graph problem and then explain the background that motivates our examination.

Problem Statement. Let $G = (V, E)$ be a simple (no self-loops and no multi-edges), connected, undirected graph, with $|V| = n \geq 1$ vertices and $|E| = m$ edges. One version of the *reassembling* of G is edge-directed and can be defined by a *total order* Θ of the m edges of G . Informally and very simply, a total order Θ of the edges gives rise to a reassembling of G as follows:

- (1) We cut every edge into two halves, thus obtaining a collection of n disconnected one-vertex components, such that for every $v \in V$ the one-vertex component $\{v\}$ has $|degree(v)|$ half edges attached to it.
- (2) We reconnect the two halves of every edge in the order specified by Θ , obtaining larger and larger components, until the original G is fully reassembled.

*Partially supported by NSF awards CCF-0820138 and CNS-1135722.

To distinguish this reassembling of G according to an order Θ from a later reassembling of G more suitable for parallel computation, we call the former *sequential reassembling* and the latter *binary reassembling*.

A *bridge* is a yet-to-be-reconnected edge between two components, say, A and B , with disjoint sets of vertices; we call such components *clusters*.¹ The set of bridges between A and B is denoted $\partial(A, B)$. For technical reasons, when we reconnect one of the bridges in $\partial(A, B)$, we also reconnect all the other bridges in $\partial(A, B)$ and cross them out from further consideration in Θ . Thus, in the reassembling of G according to Θ , there are at most m steps, rather than exactly m steps.

In the case when $B = V - A$, the set $\partial(A, B)$ is the same as the cut-set of edges determined by the cut $(A, V - A)$. Instead of $\partial(A, V - A)$, we write $\partial(A)$. The *edge-boundary degree* of a cluster A is the number of bridges with only one endpoint in A , i.e., $|\partial(A)|$.

Several natural optimization problems can be associated with graph reassembling. Two such optimizations are the following, which we identify by the letters α and β throughout:

(α) Minimize the **maximum** edge-boundary degree encountered during reassembling.

(β) Minimize the **sum** of all edge-boundary degrees encountered during reassembling.

Initially, before we start reassembling, we always set the α -measure M_α to the **maximum** of all the vertex degrees, i.e., $\max \{degree(v) \mid v \in V\}$, and we set the β -measure M_β to the **sum** of the vertex degrees, i.e., $\sum \{degree(v) \mid v \in V\}$, regardless of which strategy, i.e., the order Θ of edges, is selected for the reassembling. During reassembling, after we merge disjoint nonempty clusters A and B , we update the α -measure M_α to: $\max \{M_\alpha, |\partial(A \cup B)|\}$, and the β -measure M_β to: $(M_\beta + |\partial(A \cup B)|)$. The reassembling process terminates when only one cluster remains, which is also the set V of all the vertices.

In what we call the *binary reassembling* of G , we reconnect bridges in several non-overlapping pairs of clusters simultaneously. That is, at every step – which we may call a *parallel step* for emphasis – we choose $k \geq 1$ and choose k cluster pairs $(A_1, B_1), \dots, (A_k, B_k)$, where $A_1, B_1, \dots, A_k, B_k$ are $2k$ pairwise disjoint clusters (i.e., with pairwise disjoint subsets of vertices), and simultaneously reconnect all the bridges in $\sum_{1 \leq i \leq k} \partial(A_i, B_i)$. The subsets $A_1, B_1, \dots, A_k, B_k$ may or may not include all of the vertices, i.e., in general $\sum_{1 \leq i \leq k} (A_i \uplus B_i) \subseteq V$ rather than $= V$. (We write “ \uplus ” to denote *disjoint union*.)

A binary reassembling is naturally viewed as vertex-directed and described by a binary tree \mathcal{B} – root at the top, leaves at the bottom – with n leaf nodes, one for each of the initial one-vertex clusters. Each non-leaf node in \mathcal{B} is a cluster $(A \uplus B)$ obtained by reconnecting all the bridges in $\partial(A, B)$ between the sibling clusters A and B . The first parallel step, $i = 0$, starts at the bottom in the reassembling process, by considering the n leaf nodes of \mathcal{B} and calculating the max (for α optimization) or the sum (for β optimization) of all vertex degrees. If h is the height of \mathcal{B} , the last parallel step is $i = h$, which corresponds to the root node of \mathcal{B} (the entire set V of vertices) and produces the final α -measure and β -measure. Clearly, $\lceil \log n \rceil \leq h \leq n - 1$.

Every sequential reassembling of G can be viewed as a binary reassembling of G where, at every step, only one cluster pair (A, B) is selected and one nonempty set of bridges $\partial(A, B)$ is reconnected. Conversely, by serializing (or sequencializing) parallel steps, every binary reassembling which we call *strict* can be re-defined as a sequential reassembling. Details of the correspondence between sequential and binary reassemblies are in Appendix A.

A binary reassembling is *strict* if the merging of a cluster pair (A, B) is restricted to the case $\partial(A, B) \neq \emptyset$. If an α -optimal (resp. β -optimal) binary reassembling is strict, then its serialization is an α -optimal (resp. β -optimal) sequential reassembling.

The Linear Case. A possible and natural variation (or restriction) of graph reassembling is one which we call *linear*. If, at every step of the reassembling process, we require that the cluster pair (A, B) to be merged is such that one of the two clusters, A or B (or both at the first step), is a singleton set, then the resulting reassembling is

¹These terms (*bridge*, *cluster*, and others, later in this report) are overloaded in graph-theoretical problems. We make our own use of these terms, and state it explicitly when our meaning is somewhat at variance with that elsewhere in the literature.

linear. The binary tree \mathcal{B} describing a linear reassembling of a graph G with n vertices is therefore a degenerate tree of height $h = n - 1$.

Clearly, there can be no non-trivial parallel step which merges two (or more) cluster pairs in linear reassembling, *i.e.*, no step which simultaneously merges disjoint cluster pairs (A_1, B_1) and (A_2, B_2) to form the non-singleton clusters $(A_1 \uplus B_1)$ and $(A_2 \uplus B_2)$, before they are merged to form the cluster $((A_1 \uplus B_1) \uplus (A_2 \uplus B_2))$.

Another useful way of understanding a linear reassembling of graph G is in its sequential formulation (when the reassembling is strict): The order Θ for reconnecting the edges is such that the next edge to be reconnected is always adjacent to an already re-reconnected edge, which enforces the requirement that the next cluster pair (A, B) to be merged is always such that A or B is a singleton set.

There are other natural variations of graph reassembling, such as *balanced reassembling*, whose binary tree \mathcal{B} maximizes the merging of cluster pairs at every parallel step and whose height h is therefore $\lceil \log n \rceil$. We study these in follow-up reports.

Background and Motivation. Besides questions of optimization and the variations which it naturally suggests, graph *reassembling* (and the related operation of graph *assembling*, not considered in this paper) is part of the execution by programs in a domain-specific language (DSL) for the design of flow networks [2, 7, 8, 15]. In network *reassembling*, the network is taken apart and reassembled in an order determined by the designer; in network *assembling*, the order in which components are put together is pre-determined, which is the order in which components become available to the designer.

A flow network is a directed graph where vertices and edges are assigned various attributes that regulate flow through the network.² Programs for flow-network design are meant to connect network components in such a way that *typings at their interfaces*, *i.e.*, formally specified properties at their common boundaries, are satisfied. Network typings guarantee there are no conflicting data types when different components are connected, and insure that desirable properties of safe and secure operation are not violated by these connections, *i.e.*, they are *invariant properties* of the whole network construction.

A typing τ for a *network component* A (or *cluster* A in this report's terminology) formally expresses a constraining relationship between the variables denoting the outer ports of A (or the edge-boundary $\partial(A)$ in this report). The smaller the set of outer ports of A is, the easier it is to formulate the typing τ and to test whether it is compatible with the typing τ' of another network component A' . Although every outer port of A is directed, as input port or output port, the complexity of the formulation of τ depends only on the number of outer ports (or $|\partial(A)|$ in this report), not on their directions.

If k is a uniform upper bound on the number of outer ports of all network components, the time complexity of reassembling the network without violating any component typing τ can be made linear in the size n of the completed network and exponential in the bound k – not counting the preprocessing time $f(n)$ to determine an appropriate reassembling order. Hence, the smaller are k and $f(n)$, the more efficient is the construction of the entire network. From this follows the importance of minimizing the preprocessing time $f(n)$ for finding a reassembling strategy that also minimizes the bound k .

Main Results. In this report, we restrict attention to the linear case of graph reassembling, which is interesting and natural in its own right. We first prove that α -optimization and β -optimization of linear reassembling are both NP-hard problems. We obtain these results by showing that:

- α -optimization of linear reassembling and *minimum-cutwidth linear arrangement* (CutWidth) are polynomially-reducible to each other,
- β -optimization of linear reassembling and *minimum-cost linear arrangement* (MinArr) are polynomially-reducible to each other.

²Such networks are typically more complex than the capacited directed graphs that algorithms for max-flow (and other related quantities) and its generalizations (*e.g.*, multicommodity max-flow) operate on.

Both CutWidth and MinArr have been extensively studied: They are both NP-hard in general, but, in a few non-trivial special cases, amenable to low-degree polynomial-time solutions [13]. By our polynomial-reducibility results, these polynomial-time solutions are transferable to the α -optimization and β -optimization of linear reassembling. This leaves open the problem of identifying classes of graphs, whether of practical or theoretical significance, for which there are low-degree polynomial-time solutions for our two optimization problems.

Organization of the Report. In Section 2 we give precise formal definitions of several notions underlying our entire examination. The formulation of some of these (e.g., our definition of binary trees) is not standard, but which we purposely choose in order to facilitate the subsequent analysis.

In Section 3 we give several examples to illustrate notions discussed in this Introduction, in Section 2, as well as in later sections.

In Sections 4 and 5 we prove our NP-hardness results about α -optimization and β -optimization. Some of the long technical proofs for these sections are delayed to Appendix B.

In the final short Section 6, we point to open problems and to further current research on α -optimization and β -optimization of graph reassembling.

2 Formal Definitions and Preliminary Lemmas

Let $G = (V, E)$ be a simple (no self-loops and no multi-edges) undirected graph, with $|V| = n \geq 1$ and $|E| = m \geq 1$. Throughout, there is no loss of generality in assuming that G is connected.

As pointed out in Section 1, there are two distinct definitions of the reassembling of G . The first is easier to state informally: This is the definition of *sequential reassembling*. The second definition, *binary reassembling*, is more convenient for the optimization problems we want to study. The analysis in this report and follow-up reports is based on the formal definition of binary reassembling and its variations; we delay a formal definition of sequential reassembling to Appendix A, where we also sketch the proof of the equivalence of the two definitions when reassembling is *strict* (see Definition 5).

2.1 Binary Graph-Reassembling

Our notion of a *binary reassembling* of graph G presupposes the notion of a *binary tree* over a finite set $V = \{v_1, \dots, v_n\}$. Our definition of *binary trees* is not standard, but is more convenient for our purposes.³

Definition 1 (Binary trees). An (*unordered*) *binary tree* \mathcal{B} over $V = \{v_1, \dots, v_n\}$ is a collection of non-empty subsets of V satisfying three conditions:

1. For every $v \in V$, the singleton set $\{v\}$ is in \mathcal{B} .
2. The full set V is in \mathcal{B} .
3. For every $X \in \mathcal{B}$, there is a unique $Y \in \mathcal{B}$ such that: $X \cap Y = \emptyset$ and $(X \cup Y) \in \mathcal{B}$.

The *leaf nodes* of \mathcal{B} are the singleton sets $\{v\}$, and the *root node* of \mathcal{B} is the full set V . Depending on the context, we may refer to the members of \mathcal{B} as its *nodes* or as its *clusters*. Several expected properties of \mathcal{B} , reproducing familiar ones of a standard definition, are stated in the next two propositions. \square

Proposition 2 (Properties of binary trees). *Let \mathcal{B} be a binary tree as in Definition 1, let $v \in V$, and let:*

$$(\dagger) \quad \{v\} = X_0 \subsetneq X_1 \subsetneq X_2 \subsetneq \dots \subsetneq X_p = V$$

³A standard definition of a binary tree T makes T a subset of the set of finite binary strings $\{0, 1\}^*$ such that:

- T is prefix-closed, i.e., if $t \in T$ and u is a prefix of t , then $u \in T$.
- Every node has two children, i.e., $t0 \in T$ iff $t1 \in T$ for every $t \in \{0, 1\}^*$.

The *root node* of T is the empty string ε , and a *leaf node* of T is a string $t \in T$ without children, i.e., both $t0 \notin T$ and $t1 \notin T$.

be a maximal sequence of nested clusters from \mathcal{B} . We then have:

1. The sequence in (\dagger) is uniquely defined, i.e., every maximal nested sequence starting from $\{v\}$ is the same.
2. For every cluster $Y \in \mathcal{B}$, if $\{v\} \subseteq Y$, then $Y \in \{X_0, \dots, X_p\}$.
3. There are pairwise disjoint clusters $\{Y_0, \dots, Y_{p-1}\} \subseteq \mathcal{B}$ such that, for every $0 \leq i < p$:

$$X_i \cap Y_i = \emptyset \quad \text{and} \quad X_i \cup Y_i = X_{i+1}.$$

Based on this proposition, we use the following terminology:

- We call the sequence in (\dagger) , which is unique by part 1, the *path* from the leaf node $\{v\}$ to the root node V .
- Every cluster containing v is one of the nodes along this unique path, according to part 2.
- In part 3, we call Y_i the unique *sibling* of X_i , whose unique common *parent* is X_{i+1} , for every $0 \leq i < p$.

Proof. Part 1 is a consequence of the third condition in Definition 1, which prevents any cluster/node in \mathcal{B} from having two distinct parents.

For part 2, consider the nested sequence $\{v\} \subseteq Y \subseteq V$, and extend it to a maximal nested sequence, which is uniquely defined by part 1. This implies Y is one of the nodes along the path from $\{v\}$ to the root V .

Part 3 is another consequence of the third condition in Definition 1: For every $0 \leq i < p$, Y_i is the unique cluster in \mathcal{B} such that $X_i \cap Y_i = \emptyset$ and $X_i \cup Y_i \in \mathcal{B}$. Remaining details omitted. \square

Proposition 3 (Properties of binary trees). *Let \mathcal{B} be a binary tree as in Definition 1. We then have:*

1. For all clusters $X, Y \in \mathcal{B}$, if $X \cap Y \neq \emptyset$ then $X \subseteq Y$ or $Y \subseteq X$.
2. For every cluster $X \in \mathcal{B}$, the sub-collection of clusters $\mathcal{B}_X := \{Y \in \mathcal{B} \mid Y \subseteq X\}$ is a binary tree over X , with root node X .
3. \mathcal{B} is a collection of $(2n - 1)$ clusters.

Proof. For part 1, let $Z = X \cap Y \neq \emptyset$ and consider the maximal sequence of nested clusters which extends $Z \subseteq X \subseteq V$ or $Z \subseteq Y \subseteq V$. By part 1 of Proposition 2, both X and Y must occur along this nested sequence.

For part 2, we directly check that \mathcal{B}_X satisfies the three defining properties of a binary tree. Straightforward details omitted.

Part 3 is proved by induction on $n \geq 1$. For the induction hypothesis, we assume the statement is true for every binary tree with less than n leaf nodes, and we then prove that the statement is true for an arbitrary binary tree over V with $|V| = n$. Consider a largest cluster $X \in \mathcal{B}$ such that $X \neq V$. There is a unique $Y \in \mathcal{B}$ such that $X \cap Y = \emptyset$ and $X \cup Y \in \mathcal{B}$. Because X is largest in size but smaller than V , it must be that $X \cup Y = V$, which implies that $\{X, Y\}$ is a two-block partition of V . Consider now the subtrees \mathcal{B}_X and \mathcal{B}_Y , apply the induction hypothesis to each, and draw the desired conclusion. \square

A common measure for a standard definition of binary trees is *height*, which becomes for our notion of binary trees in Definition 1:

$$\text{height}(\mathcal{B}) := \max \left\{ p \mid \text{there is } v \in V \text{ such that} \right. \\ \left. \{v\} = X_0 \subsetneq X_1 \subsetneq \dots \subsetneq X_p = V \text{ is a maximal sequence of nested clusters} \right\}.$$

For a particular node/cluster $X \in \mathcal{B}$, the subtree of \mathcal{B} rooted at X is \mathcal{B}_X , by part 2 in Proposition 3. The height of X in \mathcal{B} is therefore $\text{height}_{\mathcal{B}}(X) := \text{height}(\mathcal{B}_X)$.

If $X, Y \subseteq V$ are disjoint sets of vertices, $\partial_G(X, Y)$ is the subset of edges of G with one endpoint in each of X and Y . If $Y = V - X$, we write $\partial_G(X)$ instead of $\partial_G(X, V - X)$.

Definition 4 (*Binary reassembling*). A *binary reassembling* of the graph $G = (V, E)$ is simply defined by a pair (G, \mathcal{B}) where \mathcal{B} is a binary tree over V , as in Definition 1.

Given a binary reassembling (G, \mathcal{B}) of G , two measures are of particular interest for our later analysis, namely, for every cluster $X \in \mathcal{B}$, the *degree* of X and the *height* of X :

$$\text{degree}_{G, \mathcal{B}}(X) := |\partial_G(X)| \quad \text{and} \quad \text{height}_{G, \mathcal{B}}(X) := \text{height}_{\mathcal{B}}(X).$$

If the context makes clear the binary reassembling (G, \mathcal{B}) – respectively, the binary tree \mathcal{B} – relative to which these measures are defined, we write $\text{degree}(X)$ and $\text{height}(X)$ – respectively, $\text{degree}_G(X)$ and $\text{height}_G(X)$ – instead of $\text{degree}_{G, \mathcal{B}}(X)$ and $\text{height}_{G, \mathcal{B}}(X)$.⁴ \square

Definition 5 (*Strict binary reassembling*). We say the binary reassembling (G, \mathcal{B}) in Definition 4 is *strict* if it satisfies the following condition: For all clusters $X, Y \in \mathcal{B}$, if X and Y are sibling nodes, then $\partial_G(X, Y) \neq \emptyset$. (In Definition 4, when we merge sibling clusters $X, Y \in \mathcal{B}$, we do not require that $\partial_G(X, Y) \neq \emptyset$.) \square

2.2 Optimization Problems

The following definition repeats a definition in Section 1 more formally.

Definition 6 (*Measures on the reassembling of a graph*). Let (G, \mathcal{B}) be a binary reassembling of G . We define the measures α and β on (G, \mathcal{B}) as follows:

$$\alpha(G, \mathcal{B}) := \max \{ \text{degree}_{G, \mathcal{B}}(X) \mid X \in \mathcal{B} \},$$

$$\beta(G, \mathcal{B}) := \sum \{ \text{degree}_{G, \mathcal{B}}(X) \mid X \in \mathcal{B} \}.$$

An optimization problem arises with the minimization of each of these measures. For example, the optimal α -measure of graph G is:

$$\alpha(G) = \min \{ \alpha(G, \mathcal{B}) \mid (G, \mathcal{B}) \text{ is a binary reassembling of } G \}.$$

We say the binary reassembling (G, \mathcal{B}) is α -*optimal* iff $\alpha(G, \mathcal{B}) = \alpha(G)$. We leave to the reader the obvious similar definition of what it means for the binary reassembling (G, \mathcal{B}) to be β -*optimal*. \square

2.3 Linear Graph-Reassembling

Let $G = (V, E)$ be a simple undirected graph with $|V| = n$. We say the binary reassembling (G, \mathcal{B}) is *linear* if \mathcal{B} is a *linear binary tree* over V , i.e., all the clusters of size ≥ 2 forms a single nested chain of length $(n - 1)$:

$$(A) \quad X_1 \subsetneq X_2 \subsetneq \dots \subsetneq X_{n-1} = V$$

This implies the height of \mathcal{B} is $(n - 1)$. By part 3 in Proposition 2, there are n leaf nodes/singleton clusters $\{Y_0, \dots, Y_{n-1}\} \subseteq \mathcal{B}$ such that $X_1 = Y_0 \cup Y_1$ and for every $1 \leq i \leq n - 2$:

$$(B) \quad X_i \cap Y_{i+1} = \emptyset \quad \text{and} \quad X_i \cup Y_{i+1} = X_{i+1}.$$

We use the letter \mathcal{L} to denote a linear binary tree, and write (G, \mathcal{L}) to denote a linear reassembling of G .

In Definition 7, we mostly use the notation and conventions of [13] and the references therein. We write \overline{vw} to denote the edge connecting vertex v and vertex w .

⁴ Our binary reassembling of G can be viewed as the “hierarchical clustering” of G , similar to a method of analysis in data mining, though used for a different purpose. Our binary reassembling mimicks what is called “agglomerative, or bottom-up, hierarchical clustering” in data mining.

Definition 7 (*Linear arrangements and cutwidths*). A linear arrangement φ of the graph $G = (V, E)$, where $|V| = n$, is a bijection $\varphi : V \rightarrow \{1, \dots, n\}$. We refer to this linear arrangement by writing (G, φ) .

Following [13], given linear arrangement (G, φ) and $i \in \{1, \dots, n\}$, we define a two-block partition of the vertices, $V = L(G, \varphi, i) \uplus R(G, \varphi, i)$, where:

$$L(G, \varphi, i) := \{v \in V \mid \varphi(v) \leq i\} \quad \text{and} \quad R(G, \varphi, i) := \{w \in V \mid \varphi(w) > i\}.$$

The *edge cut at position i* , denoted $\zeta(G, \varphi, i)$, is the number of edges connecting $L(G, \varphi, i)$ and $R(G, \varphi, i)$:

$$\zeta(G, \varphi, i) := \left| \{ \overline{vw} \in E \mid v \in L(G, \varphi, i) \text{ and } w \in R(G, \varphi, i) \} \right|.$$

In our notation in Definition 4, we have:

$$\zeta(G, \varphi, i) = \left| \partial(L(G, \varphi, i), R(G, \varphi, i)) \right| = \text{degree}(L(G, \varphi, i)).$$

The *length of \overline{vw} in the linear arrangement (G, φ)* , denoted $\xi(G, \varphi, \overline{vw})$, is “1 + the number of vertices between v and w ”:

$$\xi(G, \varphi, \overline{vw}) := |\varphi(v) - \varphi(w)|.$$

The α -measure and β -measure of the linear arrangement (G, φ) are defined by:

$$\begin{aligned} \alpha(G, \varphi) &:= \max \{ \zeta(G, \varphi, i) \mid 1 \leq i \leq n \} = \max \{ \text{degree}(L(G, \varphi, i)) \mid 1 \leq i \leq n \}, \\ \beta(G, \varphi) &:= \sum \{ \zeta(G, \varphi, i) \mid 1 \leq i \leq n \} = \sum \{ \text{degree}(L(G, \varphi, i)) \mid 1 \leq i \leq n \}. \end{aligned}$$

In the literature, $\alpha(G, \varphi)$ is called the *cutwidth* of the linear arrangement (G, φ) . The *cost* of the linear arrangement (G, φ) is usually defined as the total length of all the edges relative to (G, φ) , i.e., the cost is the measure $\gamma(G, \varphi)$ given by:

$$\gamma(G, \varphi) := \sum \{ \xi(G, \varphi, \overline{vw}) \mid \overline{vw} \in E \}.$$

However, by Lemma 8 below, $\beta(G, \varphi)$ is equal to $\gamma(G, \varphi)$. □

Lemma 8. For every linear arrangement (G, φ) , we have $\beta(G, \varphi) = \gamma(G, \varphi)$.

Proof. We have to prove that

$$\sum \{ \zeta(G, \varphi, i) \mid 1 \leq i \leq n \} = \sum \{ \xi(G, \varphi, \overline{vw}) \mid \overline{vw} \in E \}.$$

This equality holds whether G is connected or not. So, a formal proof (omitted) can be written by induction on the number $m \geq 0$ of edges in G . But informally, for every edge $\overline{vw} \in E$, if $\varphi(v) = i$ and $\varphi(w) = j$ with $i < j$, then its length $\xi(G, \varphi, \overline{vw}) = j - i$. In this case, the length of edge \overline{vw} contributes one unit to each of $(j - i)$ consecutive edge cuts: $\zeta(G, \varphi, i), \dots, \zeta(G, \varphi, j - 1)$. Hence, if we delete edge \overline{vw} from the graph, we decrease the two quantities:

$$\sum \{ \zeta(G, \varphi, i) \mid 1 \leq i \leq n \} \quad \text{and} \quad \sum \{ \xi(G, \varphi, \overline{vw}) \mid \overline{vw} \in E \}.$$

by exactly the same amount $(j - i)$. The desired conclusion follows. □

Definition 9 (*Optimal linear arrangements*). Let $G = (V, E)$ be a simple undirected graph. We say the linear arrangement (G, φ) is α -optimal if:

$$\alpha(G, \varphi) = \min \{ \alpha(G, \varphi') \mid (G, \varphi') \text{ is a linear arrangement} \}.$$

The α -optimal linear arrangement problem is the problem of defining a bijection $\varphi : V \rightarrow \{1, \dots, n\}$ such that (G, φ) is α -optimal. We define similarly the β -optimal linear arrangement problem. □

Definition 10 (*Linear arrangement induced by linear reassembling*). Let $G = (V, E)$ be a simple undirected graph and (G, \mathcal{L}) be a linear reassembling of G . Using the notation in (A) and (B) in the opening paragraph of Section 2.3, the n leaf nodes (or singleton clusters) of \mathcal{L} are: $Y_0, Y_1, Y_2, \dots, Y_{n-1}$. Observe that the order of the singletons Y_2, \dots, Y_{n-1} and, therefore, the order of the $(n-2)$ vertices in $Y_2 \cup \dots \cup Y_{n-1}$ is uniquely determined by the chain in (A), but this is not the case for the order in which we write Y_0 and Y_1 , i.e., the first cluster X_1 in (A) is equal to both $Y_0 \cup Y_1$ and $Y_1 \cup Y_0$.

We want to extract a linear arrangement (G, φ) from the linear reassembling (G, \mathcal{L}) . This is achieved by defining $\varphi : V \rightarrow \{1, \dots, n\}$ as follows:

- Let $Y_0 = \{v\}$ and $Y_1 = \{v'\}$.
 - If $\text{degree}(v) \leq \text{degree}(v')$ then set $\varphi(v) := 1$ and $\varphi(v') := 2$.
 - If $\text{degree}(v) \geq \text{degree}(v')$ then set $\varphi(v') := 1$ and $\varphi(v) := 2$.
- For every $2 \leq i \leq n-1$, if $Y_i = \{v\}$ then set $\varphi(v) := i+1$.

It is possible that $\text{degree}(v) = \text{degree}(v')$, in which case φ may place v first and v' second, or v' first and v second. This ambiguity is harmless for our analysis, in that it does not affect the α -measure and the β -measure of the linear arrangement (G, φ) .

Whether φ places v first and v' second, or v' first and v second, we call (G, φ) a linear arrangement of G induced by the linear reassembling (G, \mathcal{L}) .

Note that, for every $1 \leq i \leq n-1$, we have the equality $X_i = L(G, \varphi, i)$, where $L(G, \varphi, i)$ is the set of vertices at position i and to the left of it, as in Definition 7. \square

Definition 11 (*Linear reassembling induced by linear arrangement*). Let $G = (V, E)$ be a simple undirected graph and let (G, φ) be a linear arrangement of G . We extract a linear reassembling (G, \mathcal{L}) from (G, φ) . The n leaf nodes/singleton clusters of \mathcal{L} are:

$$\{\{v\} \mid v \in V\} = \{\{\varphi^{-1}(i)\} \mid 1 \leq i \leq n\}.$$

The $(n-1)$ non-leaf nodes/clusters of \mathcal{L} are:

$$\begin{aligned} X_1 &:= \{\varphi^{-1}(1), \varphi^{-1}(2)\}, \\ X_2 &:= \{\varphi^{-1}(1), \varphi^{-1}(2), \varphi^{-1}(3)\}, \\ &\dots \\ X_{n-1} &:= \{\varphi^{-1}(1), \varphi^{-1}(2), \varphi^{-1}(3), \dots, \varphi^{-1}(n)\} = V. \end{aligned}$$

We call (G, \mathcal{L}) the linear reassembling of G induced by the linear arrangement (G, φ) . \square

3 Examples

We present several simple examples to illustrate notions mentioned in Sections 1 and 2. We first introduce a convenient notation for specifying binary trees over a set V of vertices. Let 2^V denote the power set of V . We define a binary operation $\overline{\mathcal{X} \mathcal{Y}}$ for all $\mathcal{X} \subseteq 2^V$ and $\mathcal{Y} \subseteq 2^V$ by:

$$\overline{\mathcal{X} \mathcal{Y}} := (\mathcal{X} \cup \mathcal{Y}) \cup ((\bigcup \mathcal{X}) \cup (\bigcup \mathcal{Y}))$$

The examples below illustrate how we use this operation “ $\overline{}$ ”.

We overload the overline notation “ $\overline{}$ ” to denote a nonempty subset of V , i.e., $\overline{v_1 \dots v_k}$ means $\{v_1, \dots, v_k\}$. In particular, the edge connecting v and w is denoted by the two-vertex set $\overline{vw} = \{v, w\}$.

The context will make clear whether we use “ $\overline{}$ ” to refer to a set of subsets of V (in the case of binary trees) or to just a subset of V .

Example 12. The hypercube graph Q_3 is shown on the left in Figure 1, and three of its reassemblings are shown on the right in Figure 1. The top reassembling is neither balanced nor linear; the middle one is *balanced*; and the bottom one is *linear*.

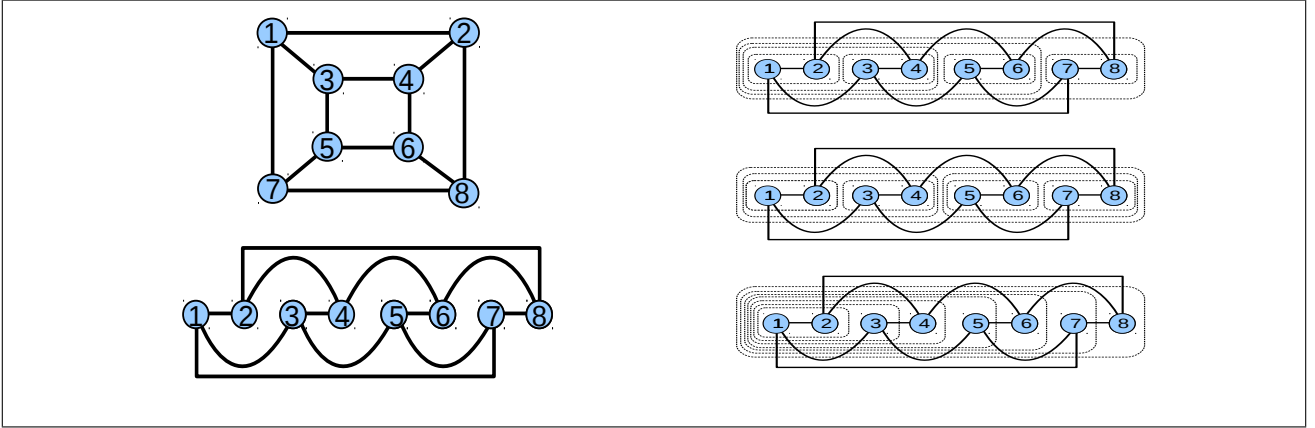


Figure 1: Two different drawings of the hypercube graph Q_3 (on the left) and three of its reassemblings (on the right).

For each of the three reassemblings on the right in Figure 1, from top to bottom, we list the unique binary tree \mathcal{B} over the vertices $\{1, \dots, 8\}$ underlying it (in its *binary reassembling* formulation) and one of the orderings Θ of the edges $\{\overline{12}, \dots, \overline{78}\}$ inducing it (in its *sequential reassembling* formulation):

$$\begin{aligned}
 \mathcal{B}_1 &= \left\{ \overline{\overline{\overline{1\ 2\ 3\ 4\ 5\ 6\ 7\ 8}}} \right\} & \Theta_1^{Q_3} &= \overline{1\ 2} \ \overline{3\ 4} \ \overline{5\ 6} \ \overline{7\ 8} \ \overline{1\ 3} \ \overline{3\ 5} \ \overline{5\ 7} \ \dots \\
 \mathcal{B}_2 &= \left\{ \overline{\overline{\overline{1\ 2\ 3\ 4} \ \overline{5\ 6\ 7\ 8}}} \right\} & \Theta_2^{Q_3} &= \overline{1\ 2} \ \overline{3\ 4} \ \overline{5\ 6} \ \overline{7\ 8} \ \overline{1\ 3} \ \overline{5\ 7} \ \overline{3\ 5} \ \dots \quad (\text{balanced}) \\
 \mathcal{B}_3 &= \left\{ \overline{\overline{\overline{\overline{\overline{1\ 2\ 3\ 4\ 5\ 6\ 7\ 8}}}}} \right\} & \Theta_3^{Q_3} &= \overline{1\ 2} \ \overline{1\ 3} \ \overline{3\ 4} \ \overline{3\ 5} \ \overline{5\ 6} \ \overline{5\ 7} \ \overline{7\ 8} \ \dots \quad (\text{linear})
 \end{aligned}$$

where, for simplicity, we write just “ v ” instead of the cumbersome $\{\{v\}\} = \overline{\overline{v}}$. Thus, for example, two of the subsets of \mathcal{B}_1 above appear as “ $\overline{1\ 2}$ ” and “ $\overline{\overline{1\ 2\ 3\ 4}}$ ”, and if we expand them in full, we obtain:

$$\overline{1\ 2} = \{\{1\}, \{2\}, \{1, 2\}\} \quad \text{and} \quad \overline{\overline{1\ 2\ 3\ 4}} = \{\{1\}, \{2\}, \{3\}, \{4\}, \{1, 2\}, \{3, 4\}, \{1, 2, 3, 4\}\}.$$

The ellipsis “ \dots ” in the definition of $\Theta_i^{Q_3}$ above are the remaining edges of Q_3 , which can be listed in any order without changing the reassembling.⁵ A simple calculation of the α -measure and β -measure of these three reassemblings of Q_3 produces:

$$\begin{aligned}
 \alpha(Q_3, \mathcal{B}_1) &= \alpha(Q_3, \mathcal{B}_2) = 4 \quad \text{and} \quad \alpha(Q_3, \mathcal{B}_3) = 5, \\
 \beta(Q_3, \mathcal{B}_1) &= \beta(Q_3, \mathcal{B}_2) = 48 \quad \text{and} \quad \beta(Q_3, \mathcal{B}_3) = 49.
 \end{aligned}$$

By exhaustive inspection (details omitted), (Q_3, \mathcal{B}_1) is both α -optimal and β -optimal for the class of all binary reassemblings of Q_3 . Because the α -measure and β -measure of (Q_3, \mathcal{B}_1) and those of (Q_3, \mathcal{B}_2) are equal,

⁵ We qualify $\Theta_i^{Q_3}$ with the superscript “ Q_3 ” because it depends on the graph Q_3 . The same ordering of the edges may not be valid for a sequential reassembling of another 8-vertex graph with a set of edges different from that of Q_3 . This is not the case for the binary tree \mathcal{B} underlying the binary reassembling (G, \mathcal{B}) of a graph $G = (V, E)$; that is, regardless of the placement of edges in G , the tree \mathcal{B} over V is valid for the binary reassembling (G, \mathcal{B}) and again for the binary reassembling (G', \mathcal{B}) of every graph $G' = (V, E')$ over the same set V of vertices.

(Q_3, \mathcal{B}_2) is also both α -optimal and β -optimal for the class of all binary reassemblings and, therefore, for the smaller class of all *balanced* reassemblings of which it is a member. By exhaustive inspection again, (Q_3, \mathcal{B}_3) is both α -optimal and β -optimal for the sub-class of all *linear* reassemblings, but not for the full class of all binary reassemblings of Q_3 . \square

Example 13. The complete graph K_8 on 8 vertices is shown in Figure 2. We can carry out three reassemblings of K_8 by using the binary trees \mathcal{B}_1 , \mathcal{B}_2 , and \mathcal{B}_3 , from Example 12 again.

A straightforward calculation of the α -measure and β -measure of the resulting reassemblings (K_8, \mathcal{B}_1) , (K_8, \mathcal{B}_2) , and (K_8, \mathcal{B}_3) produces the following values:

$$\begin{aligned}\alpha(K_8, \mathcal{B}_1) &= \alpha(K_8, \mathcal{B}_2) = \alpha(K_8, \mathcal{B}_3) = 16, \\ \beta(K_8, \mathcal{B}_1) &= 132, \quad \beta(K_8, \mathcal{B}_2) = 136, \quad \beta(K_8, \mathcal{B}_3) = 133.\end{aligned}$$

Because of the symmetries of K_8 (“every permutation of the 8 vertices produces another graph isomorphic to K_8 ”), all *balanced* reassemblings are isomorphic and so are all *linear* reassemblings. Hence, (K_8, \mathcal{B}_2) is trivially α -optimal and β -optimal for the class of all balanced reassemblings of K_8 , and (K_8, \mathcal{B}_3) is trivially α -optimal and β -optimal for the class of all linear reassemblings of K_8 .

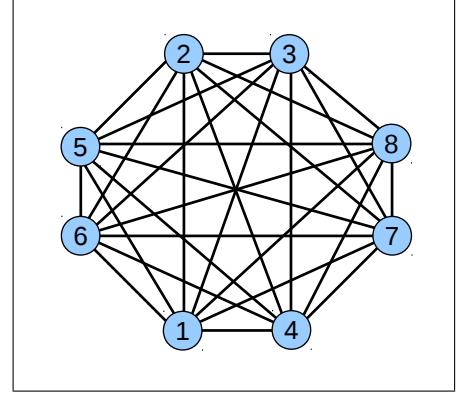


Figure 2: Complete graph K_8 .

By exhaustive inspection (details omitted), it turns out that (K_8, \mathcal{B}_1) is α -optimal for the class of all binary reassemblings, but it is not β -optimal for the same class. The underlying tree of a β -optimal binary reassembling of K_8 turns out to be the following \mathcal{B}_4 over the vertices $\{1, \dots, 8\}$, shown with an ordering $\Theta_4^{K_8}$ of the edges which induces a sequential ordering equal to (K_8, \mathcal{B}_4) :

$$\mathcal{B}_4 = \left\{ \begin{array}{c} \overline{\overline{\overline{\overline{1 \ 2 \ 3 \ 4} \ 5 \ 6} \ 7 \ 8}} \end{array} \right. \quad \Theta_4^{K_8} = \overline{1 \ 2} \ \overline{3 \ 4} \ \overline{5 \ 6} \ \overline{1 \ 3} \ \overline{1 \ 5} \ \overline{1 \ 7} \ \overline{1 \ 8} \ \dots$$

where the ellipsis “...” are the remaining edges in any order. The resulting β -measure is $\beta(K_8, \mathcal{B}_4) = 127$. \square

Example 14. The star graph S_7 , with 7 leaves and one internal vertex, is shown in Figure 3. We can carry out four reassemblings of S_7 by using the binary trees \mathcal{B}_1 , \mathcal{B}_2 , \mathcal{B}_3 , and \mathcal{B}_4 , from Examples 12 and 13 again.

From a straightforward calculation of the α -measure and β -measure of the reassemblings (S_7, \mathcal{B}_1) , (S_7, \mathcal{B}_2) , (S_7, \mathcal{B}_3) , and (S_7, \mathcal{B}_4) :

$$\begin{aligned}\alpha(S_7, \mathcal{B}_1) &= \alpha(S_7, \mathcal{B}_2) = \alpha(S_7, \mathcal{B}_3) = \alpha(S_7, \mathcal{B}_4) = 7, \\ \beta(S_7, \mathcal{B}_1) &= 32, \quad \beta(S_7, \mathcal{B}_2) = 34, \\ \beta(S_7, \mathcal{B}_3) &= 35, \quad \beta(S_7, \mathcal{B}_4) = 31.\end{aligned}$$

The preceding four reassemblings are all α -optimal, each for its own class of reassemblings, *i.e.*, (S_7, \mathcal{B}_2) is α -optimal for the class of *balanced* reassemblings of S_7 and (S_7, \mathcal{B}_3) for the class of *linear* reassemblings of S_7 . It turns out that only (S_7, \mathcal{B}_2) is β -optimal for its own class, the class of *balanced* reassemblings of S_7 . None of the four is β -optimal for the class of all binary reassemblings of S_7 .

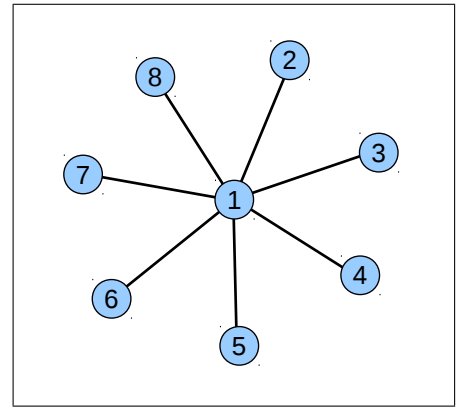


Figure 3: Star graph S_7 .

Of the four binary reassemblings above, only (S_7, \mathcal{B}_3) is *strict*; the three other are not strict, *i.e.*, the three other merge some cluster pairs (A, B) such that $\partial(A, B) = \emptyset$. Because (S_7, \mathcal{B}_1) , (S_7, \mathcal{B}_2) , and (S_7, \mathcal{B}_4) are not strict, it is not possible to re-define them as sequential reassemblings, each relative to an appropriate edge ordering. An ordering $\Theta_3^{S_7}$ of the edges that induces a sequential reassembling equal to (S_7, \mathcal{B}_3) is:

$$\Theta_3^{S_7} = \overline{1 \ 2} \ \overline{1 \ 3} \ \overline{1 \ 4} \ \overline{1 \ 5} \ \overline{1 \ 6} \ \overline{1 \ 7} \ \overline{1 \ 8}.$$

By exhaustive inspection (details omitted), the following is a binary tree \mathcal{B}_5 over the vertices $\{1, \dots, 8\}$ such that (S_7, \mathcal{B}_5) is β -optimal for the class of all binary reassemblings of S_7 . Since (S_7, \mathcal{B}_5) is not strict, there is no corresponding ordering $\Theta_5^{S_7}$ of the edges:

[illegible]

The resulting β -measure is $\beta(S_7, \mathcal{B}_5) = 29$. Note that \mathcal{B}_5 is also linear. Hence, (S_7, \mathcal{B}_5) is also β -optimal for the class of linear reassemblings of S_7 . \square

Example 15. The binary tree \mathcal{B}_3 in Examples 12, 13, and 14, is a linear binary tree. Written in full, using the notation in (A) at the beginning of Section 2.3, the non-singleton sets of \mathcal{B}_3 are:

$$\begin{aligned} X_1 &= \overline{12}, & X_2 &= \overline{123}, & X_3 &= \overline{1234}, & X_4 &= \overline{12345}, \\ X_5 &= \overline{123456}, & X_6 &= \overline{1234567}, & X_7 &= \overline{12345678}. \end{aligned}$$

The singleton sets of \mathcal{B}_3 are:

$$Y_0 = \bar{1}, \quad Y_1 = \bar{2}, \quad Y_2 = \bar{3}, \quad Y_3 = \bar{4}, \quad Y_4 = \bar{5}, \quad Y_5 = \bar{6}, \quad Y_6 = \bar{7}, \quad Y_7 = \bar{8}.$$

The linear arrangement φ_3 induced by the linear reassembling (S_7, \mathcal{B}_3) in Example 14 is (see Definition 10):

$$\varphi_3(2) = 1, \quad \varphi_3(1) = 2, \quad \varphi_3(3) = 3, \quad \varphi_3(4) = 4, \quad \varphi_3(5) = 5, \quad \varphi_3(6) = 6, \quad \varphi_3(7) = 7, \quad \varphi_3(8) = 8,$$

rather than the linear arrangement φ' :

$$\varphi'_3(1) = 1, \quad \varphi'_3(2) = 2, \quad \varphi'_3(3) = 3, \quad \varphi'_3(4) = 4, \quad \varphi'_3(5) = 5, \quad \varphi'_3(6) = 6, \quad \varphi'_3(7) = 7, \quad \varphi'_3(8) = 8,$$

because $\text{degree}_{S_7}(2) = 1 < 7 = \text{degree}_{S_7}(1)$. The difference between φ_3 and φ'_3 is in the placement of the two first vertices: vertex “1” and vertex “2”. \square

Example 16. The binary tree \mathcal{B}_5 in Example 14 is a linear binary tree. As in Example 15, it is straightforward to specify the singleton and non-singleton sets of \mathcal{B}_5 (omitted here) to fit the notation of (A) and (B) at the beginning of Section 2.3. There are two possible linear arrangements, φ_5 and φ'_5 , which are induced by the linear reassembling (S_7, \mathcal{B}_5) , because $\text{degree}_{S_7}(2) = \text{degree}_{S_7}(3) = 1$ (see Definition 10), namely:

$$\varphi_5(2) = 1, \quad \varphi_5(3) = 2, \quad \varphi_5(4) = 3, \quad \varphi_5(1) = 4, \quad \varphi_5(5) = 5, \quad \varphi_5(6) = 6, \quad \varphi_5(7) = 7, \quad \varphi_5(8) = 8,$$

$$\varphi'_5(3) = 1, \quad \varphi'_5(2) = 2, \quad \varphi'_5(4) = 3, \quad \varphi'_5(1) = 4, \quad \varphi'_5(5) = 5, \quad \varphi'_5(6) = 6, \quad \varphi'_5(7) = 7, \quad \varphi'_5(8) = 8.$$

The difference between φ_5 and φ'_5 is in the placement of the two first vertices: vertex “2” and vertex “3”. In contrast to φ_3 and φ'_3 in Example 15, both φ_5 and φ'_5 are valid as linear arrangements induced by the linear reassembling (S_7, \mathcal{B}_5) . A comparison between φ_3 and φ_5 is shown in Figure 4. \square

Example 17. This is a continuation of Example 15. The linear reassembling induced by the linear arrangement (S_7, φ_3) is precisely (S_7, \mathcal{B}_3) , but so is the linear reassembling induced by the linear arrangement (S_7, φ'_3) again the same (S_7, \mathcal{B}_3) , according to Definition 11. This means: *linear arrangements make distinction that linear reassemblings do not make*. This difference is in the placement of the two first vertices, specifically:

- The α -measure and β -measure of a *linear arrangement* generally depend on which vertex is placed first and which is placed second.
- The α -measure and β -measure of a *linear reassembling* do *not* distinguish between a first and second vertex and do *not* depend on which is placed first and which is placed second.

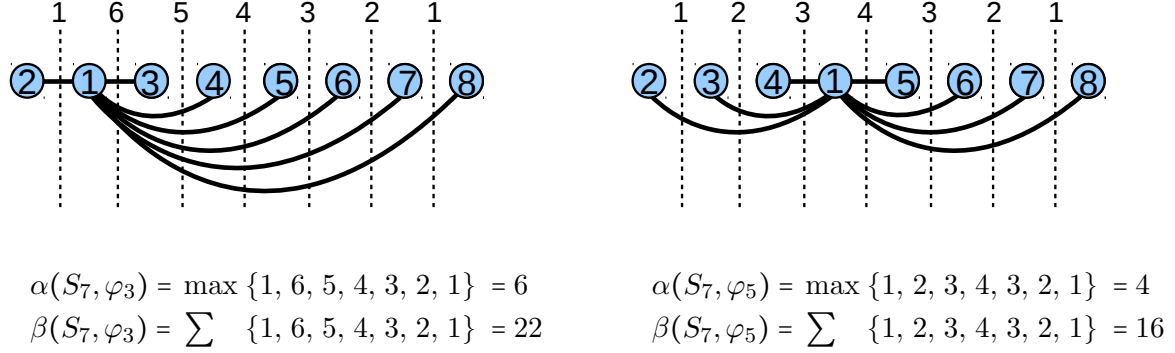


Figure 4: Comparison of linear arrangements (S_7, φ_3) in Example 15 and (S_7, φ_5) in Example 16.

As an example, consider the linear arrangements (S_7, φ_3) and (S_7, φ'_3) . Their α -measure and β -measure are:

$$\begin{aligned} \alpha(S_7, \varphi_3) &= 6, & \beta(S_7, \varphi_3) &= 22, \\ \alpha(S_7, \varphi'_3) &= 7, & \beta(S_7, \varphi'_3) &= 28. \end{aligned}$$

For the linear reassembling (S_7, \mathcal{B}_3) induced by both (S_7, φ_3) and (S_7, φ'_3) , we have: $\alpha(S_7, \mathcal{B}_3) = 7$ and $\beta(S_7, \mathcal{B}_3) = 35$, as noted in Example 14. Moreover, while both (S_7, φ_3) and (S_7, φ'_3) are neither α -optimal nor β -optimal, (S_7, \mathcal{B}_3) is α -optimal (though not β -optimal). \square

4 α -Optimization of Linear Reassembling Is NP-Hard

We prove that the α -optimality of linear arrangements (in the literature: the *minimum-cutwidth linear arrangement problem*) and the α -optimality of linear reassemblings are reducible to each other in polynomial time.

Definition 18 (*Chordal graph, triangulation, clique number, treewidth*). Let $G = (V, E)$ be a simple undirected graph. The following are standard notions of graph theory [5].

- G is a *chordal graph* if every cycle of length of 4 or more has a *chord*, i.e., an edge connecting two vertices that are not consecutive in the cycle.
- A *triangulation* of G is a chordal graph $G' = (V', E')$ where $V = V'$ and $E \subseteq E'$. In such a case, we say that G can be *triangulated into* G' , not uniquely in general.
- The *clique number* of G , denoted $\omega(G)$, is the size of a largest clique in G .
- There are different equivalent definitions of the *treewidth*. We here use a definition, or a consequence of the original definition, which is more convenient for our purposes [3, 5]. The *treewidth* of G is:

$$\min \{ \omega(G') \mid G' \text{ is a triangulation of } G \} - 1.$$

In words, among all triangulations G' of G , we choose a G' whose clique number is smallest: The *treewidth* of G is one less than the clique number of such a G' . \square

Lemma 19. *For every positive integers Δ and k , there is an algorithm \mathcal{A} using Δ and k as fixed parameters, such that, given an arbitrary simple undirected graph $G = (V, E)$ as input to \mathcal{A} , if:*

1. *the maximum vertex degree in G is $\leq \Delta$, and*
2. *the treewidth of G is $\leq k$,*

then \mathcal{A} computes a minimum-cutwidth linear arrangement of G in time $\mathcal{O}(n^{\Delta k^2})$ where $n = |V|$.

Proof. This is Theorem 4.2 in [16], where the algorithm not only computes the value of a minimum cutwidth, but can be adjusted to output the corresponding minimum-cutwidth linear arrangement $\varphi : V \rightarrow \{1, \dots, n\}$. \square

In Lemma 20, a *cut vertex* in G is a vertex whose removal increases the number of connected components.

Lemma 20. *There is an algorithm \mathcal{A} such that, given an arbitrary simple undirected graph $G = (V, E)$ as input to \mathcal{A} , if:*

1. *every vertex in G has degree ≤ 3 , and*
2. *every vertex in G of degree = 3 is a cut vertex,*

then \mathcal{A} computes a minimum-cutwidth linear arrangement of G in time $\mathcal{O}(n^{12})$ where $n = |V|$.

Proof. We show that the treewidth k of G is ≤ 2 . Because the maximum vertex degree Δ of G is ≤ 3 , Lemma 19 implies the existence of an algorithm \mathcal{A} which runs in time $\mathcal{O}(n^{\Delta k^2}) = \mathcal{O}(n^{12})$.

To show that $k \leq 2$, consider a vertex v of degree = 3, which is therefore a cut vertex. The removal of v can have one of two possible outcomes:

- (a) disconnect G into 3 components, or
- (b) disconnect G into 2 components.

If every vertex of degree = 3 satisfies condition (a), then G is tree whose treewidth is 1, since its clique number $\omega(G) = 2$ in this case.

If C_1 and C_2 are cycles in G , each with 3 vertices or more, then C_1 and C_2 are non-overlapping, i.e., C_1 and C_2 have no vertex in common and no edge in common. If they have an edge $\overline{v_1 w_1}$ in common, then there is an edge $\overline{v_2 w_2} \in C_1 \cap C_2$ such that $\text{degree}(v_2) = 3$ (or, resp., $\text{degree}(w_2) = 3$) and v_2 (or, resp., w_2) is not a cut vertex, contradicting the hypothesis. If C_1 and C_2 have no edge in common, but do have a vertex v in common, then $\text{degree}(v) > 3$, again contradicting the hypothesis.

In case one or more vertices satisfy condition (b), G can be therefore viewed as a finite collection of non-overlapping rings $\{R_1, \dots, R_p\}$, each ring being a cycle with at least 3 vertices, satisfying condition (c):

- (c) if two distinct rings $\{R_i, R_j\}$, with $i \neq j$, are connected by a path $P_{i,j}$, then the removal of all the vertices and edges of $P_{i,j}$ (in particular the two endpoints of $P_{i,j}$, one in R_i and one in R_j , which are necessarily vertices of degree = 3) disconnects G into 2 components.

Another way of expressing (c) is that, if all the rings $\{R_1, \dots, R_p\}$ are each contracted to a single vertex, then the result is a tree (where some of the internal vertices may now have degree larger than 3). Since the clique number of a ring is 3, the treewidth of a ring is 2, and the desired conclusion follows. \square

Lemma 21. *Let $G = (V, E)$ be a simple undirected graph, where every vertex has degree ≤ 3 , and let (G, \mathcal{L}) be a linear reassembling of G . Consider the longest chain of nested clusters of size ≥ 2 , as in (A) in the opening paragraph of Section 2.3:*

$$X_1 \subsetneq X_2 \subsetneq \dots \subsetneq X_{n-1} = V.$$

Conclusion: *If there is one vertex of degree = 3 in G which is not a cut vertex, then $\max \{ \text{degree}(X_1), \dots, \text{degree}(X_{n-1}) \} \geq 3$.*

Proof. We first show there are least two vertices of degree = 3 which are not cut vertices. Let v be a vertex of degree = 3 which is not a cut vertex, and let $\{\overline{vx}, \overline{vy}, \overline{vz}\}$ be the three edges incident to v . Because v is not a cut vertex, any two edges in $\{\overline{vx}, \overline{vy}, \overline{vz}\}$ are consecutive edges in a cycle containing v . Let $C(v, x, y)$ be a cycle containing edges $\{\overline{vx}, \overline{vy}\}$, and define similarly cycles $C(v, x, z)$ and $C(v, y, z)$. If any of these

three cycles contains a chord, then the two endpoints of the chord are vertices of degree = 3 which are not cut vertices. If none of these three cycles contain a chord, then we can combine any two of them, because they share an edge, to form another cycle with a chord, which again implies the existence of two vertices of degree = 3 which are not cut vertices.

To conclude the proof, consider the clusters of \mathcal{L} of size ≥ 2 : These are $\{X_1, \dots, X_{n-1}\}$, and the corresponding singleton clusters are $\{Y_0, \dots, Y_{n-1}\}$, as in (A) and (B) in Section 2.3. By the preceding argument, there are at least two vertices of degree = 3 which are not cut vertices. Let one of these two be v , with $Y_i = \{v\}$ for some $i \geq 1$.

We have $X_{i-1} \cap Y_i = \emptyset$ and $X_{i-1} \cup Y_i = X_i$. There are two cases: (1) For some vertex $w \in X_{i-1}$, there is an edge $\overline{vw} \in E$, and (2) for every vertex $w \in X_{i-1}$, there is no such edge. We consider case (1) and leave the other (easier) case (2) to the reader.

We cannot have $\text{degree}(X_{i-1}) = 0$, otherwise G is disconnected, nor can we have $\text{degree}(X_{i-1}) = 1$, otherwise v is a cut vertex. Hence, $\text{degree}(X_{i-1}) \geq 2$. If $\text{degree}(X_{i-1}) \geq 3$, this is already the conclusion of the lemma and there is nothing else to prove. Suppose $\text{degree}(X_{i-1}) = 2$, the case left to consider.

Similarly, we cannot have $\text{degree}(X_i) = 0$, otherwise G is disconnected, nor $\text{degree}(X_i) = 1$, otherwise v is a cut vertex. Hence, $\text{degree}(X_i) \geq 2$. But if $\text{degree}(X_{i-1}) = 2$ and $\text{degree}(X_i) \geq 2$, with $\text{degree}(v) = 3$, then it must be that $\text{degree}(X_i) = 3$. \square

Lemma 22. *Let $G = (V, E)$ be a simple undirected graph, where every vertex has degree ≤ 3 and where one vertex of degree = 3 is not a cut vertex. Let (G, \mathcal{L}) be a linear reassembling and (G, φ) a linear arrangement.*

Conclusion: *If (G, \mathcal{L}) is induced by (G, φ) , or if (G, φ) is induced by (G, \mathcal{L}) , then:*

- $\alpha(G, \mathcal{L}) = \alpha(G, \varphi)$.
- (G, \mathcal{L}) is α -optimal iff (G, φ) is α -optimal.

Proof. Straightforward consequence of Lemma 21, the definitions of $\alpha(G, \mathcal{L})$ and $\alpha(G, \varphi)$, and what it means for (G, \mathcal{L}) to be induced by (G, φ) and for (G, φ) to be induced by (G, \mathcal{L}) . When there is at least one vertex of degree = 3 which is *not* a cut vertex, and therefore at least two of them by the proof of Lemma 21, we can ignore the degrees of singleton clusters in the computation of $\alpha(G, \mathcal{L})$. All details omitted. \square

Theorem 23. *For the class of simple undirected graphs G where every vertex has degree ≤ 3 , the α -optimality of linear arrangements (G, φ) is polynomial-time reducible to the α -optimality of linear reassemblings (G, \mathcal{L}) .*

More explicitly, a polynomial-time algorithm \mathcal{A} , which returns an α -optimal linear reassembling (G, \mathcal{L}) of a graph G where every vertex has degree ≤ 3 , can be used to return an α -optimal linear arrangement (G, φ) .

Proof. Consider an arbitrary G where every vertex has degree ≤ 3 . If every vertex in G of degree = 3 is a cut vertex, we use the algorithm in Lemma 20 to compute an α -optimal linear arrangement (G, φ) in time $\mathcal{O}(n^{12})$. If there is a vertex in G of degree = 3 which is not a cut vertex, we first compute an α -optimal linear reassembling (G, \mathcal{L}) and then return the linear arrangement (G, φ) induced by (G, \mathcal{L}) . The desired conclusion follows from Lemma 22. \square

Corollary 24. *For the class of all simple undirected graphs G , the computation of α -optimal linear reassemblings (G, \mathcal{L}) is an NP-hard problem.*

Proof. If there is a polynomial-time algorithm \mathcal{A} to compute, for an arbitrary simple undirected graph, an α -optimal linear reassembling, then the same algorithm \mathcal{A} can be used to compute in polynomial-time an α -optimal linear reassembling (G, \mathcal{L}) for a graph G where every vertex has degree ≤ 3 . By Theorem 23, \mathcal{A} can be further adapted to compute an α -optimal linear arrangement (G, φ) for such a graph G in polynomial time. But the latter problem (in the literature: the *minimum-cutwidth linear arrangement* problem) is known to be NP-hard [10, 11]. \square

Remark 25. To the best of our knowledge, the complexity status of the *minimum-cutwidth linear arrangement* problem for k -regular graphs for a fixed $k \geq 3$ is an open problem. If it were known to be NP-hard, we would be able to simplify our proof of Theorem 23 and its corollary considerably. In particular, we would be able to eliminate Lemmas 19 and 20 and the supporting Definition 18, as well as simplify Lemmas 21 and 22 by restricting them to k -regular graphs. \square

Theorem 23 and Corollary 24 together say the α -optimality of linear arrangements (G, φ) is polynomial-time reducible to the α -optimality of linear reassemblings (G, \mathcal{L}) . For completeness, we show the converse in the next theorem.

Theorem 26. *For the class of simple undirected graphs G in general, the α -optimality of linear reassemblings (G, \mathcal{L}) is polynomial-time reducible to the α -optimality of linear arrangements (G, φ) .*

Proof. In Appendix B. \square

5 β -Optimization of Linear Reassembling Is NP-Hard

We prove that the β -optimality of linear arrangements (in the literature: the *minimum-cost linear arrangement* problem or also the *optimal linear arrangement* problem) and the β -optimality of linear reassemblings are reducible to each other in polynomial time. Towards this end, we prove an intermediate result, which is also of independent interest (Theorem 32 which presupposes Definition 27).

Definition 27 (*Anchored linear reassemblings*). Let $G = (V, E)$ be a simple undirected graph and let $w \in V$. Let (G, \mathcal{L}) be a linear reassembling of G , whose longest chain of nested clusters of size ≥ 2 , as in (A) in the opening paragraph of Section 2.3, is:

$$X_1 \subsetneq X_2 \subsetneq \dots \subsetneq X_{n-1} = V$$

and whose corresponding singleton clusters are $\{Y_0, \dots, Y_{n-1}\}$, as determined by (B) in the opening paragraph of Section 2.3. We say (G, \mathcal{L}) is a *linear reassembling anchored at $w \in V$* iff there is a vertex $w' \in V$ such that:

$$Y_0 = \{w\}, \quad Y_1 = \{w'\}, \quad \text{and } \text{degree}_G(w) \leq \text{degree}_G(w').$$

Note that we require that the immediate sibling $Y_1 = \{w'\}$ of the leaf node $Y_0 = \{w\}$ satisfy the condition $\text{degree}_G(w) \leq \text{degree}_G(w')$. This implies that, given an arbitrary vertex $w \in V$, we cannot anchor a linear reassembling at w unless we find another vertex $w' \in V$ such that $\text{degree}_G(w) \leq \text{degree}_G(w')$ and then make $\{w\}$ and $\{w'\}$ sibling leaf-nodes. This is a technical restriction to simplify the statement of Lemma 30.⁶ \square

Definition 28 (*Anchored linear arrangements*). Let $G = (V, E)$ be a simple undirected graph and let $w \in V$. Let (G, φ) be a linear arrangement of G . We say (G, φ) is a *linear arrangement anchored at $w \in V$* iff there is a vertex $w' \in V$ such that:

$$\varphi(w) = 1, \quad \varphi(w') = 2, \quad \text{and } \text{degree}_G(w) \leq \text{degree}_G(w').$$

Again, as in Definition 27, the condition $\text{degree}_G(w) \leq \text{degree}_G(w')$ is imposed in order to simplify the statement of Lemma 30. It is worth noting that, if we relax this condition and allow $\text{degree}_G(w) > \text{degree}_G(w')$, then the new arrangement φ' which permutes the positions of w and w' , i.e.:

$$\varphi'(w') = 1, \quad \varphi'(w) = 2, \quad \text{and } \varphi'(v) = \varphi(v) \text{ for all } v \in V - \{w, w'\},$$

⁶Thus, we cannot say that the linear reassembling (S_7, \mathcal{B}_3) , in Examples 14 and 15, is anchored at vertex “1”, though we can say that (S_7, \mathcal{B}_3) is anchored at vertex “2”. More generally, in the case of a star graph S_k with $k \geq 3$ leaves: There is no linear reassembling of S_k anchored at the internal vertex of S_k .

is such that $\beta(G, \varphi') < \beta(G, \varphi)$. In words, if we allowed $\text{degree}_G(w) > \text{degree}_G(w')$, the linear arrangement (G, φ) would not be β -optimal.⁷ \square

Example 29. Consider the linear reassemblings (S_7, \mathcal{B}_3) and (S_7, \mathcal{B}_5) in Example 14. (S_7, \mathcal{B}_3) is anchored at vertex “2”, but cannot be anchored at vertex “1”, while (S_7, \mathcal{B}_5) is anchored at vertex “2”, and can be anchored again at vertex “3”. Both (S_7, \mathcal{B}_3) and (S_7, \mathcal{B}_5) are α -optimal and, a fortiori, α -optimal for the class of all linear reassemblings of S_7 anchored at vertex “2”. Moreover, $\beta(S_7, \mathcal{B}_3) = 35$ and $\beta(S_7, \mathcal{B}_5) = 29$, so that (S_7, \mathcal{B}_3) is not β -optimal for the class of all linear reassemblings of S_7 anchored at vertex “2”, while (S_7, \mathcal{B}_5) is β -optimal for the same class.

Consider now the linear arrangements (S_7, φ_3) and (S_7, φ_5) induced by the linear reassemblings (S_7, \mathcal{B}_3) and (S_7, \mathcal{B}_5) , respectively. φ_3 and φ_5 are given in Example 15 and Example 16. Both (S_7, φ_3) and (S_7, φ_5) are anchored at vertex “2”. Moreover, (S_7, φ_3) cannot be anchored at vertex “1” (the sibling leaf of “2” in φ_3), while (S_7, φ_5) can be anchored again at vertex “3” (the sibling leaf of “2” in φ_5).

(S_7, φ_3) is neither α -optimal nor β -optimal for the class of all linear arrangements of S_7 anchored at “2”; hence, (S_7, φ_3) is neither α -optimal nor β -optimal for the super-class of all linear arrangements of S_7 . By contrast, (S_7, φ_5) is both α -optimal and β -optimal for the class of all linear arrangements of S_7 ; hence, (S_7, φ_5) is both α -optimal and β -optimal for the sub-class of all linear arrangements of S_7 anchored at “2”. \square

Let (G, \mathcal{L}) be a linear reassembling anchored at vertex $w \in V$. We say (G, \mathcal{L}) is β -optimal relative to anchor w iff:

$$\beta(G, \mathcal{L}) = \min \{ \beta(G, \mathcal{L}') \mid (G, \mathcal{L}') \text{ is a linear reassembling anchored at } w \}.$$

Clearly, (G, \mathcal{L}) is a β -optimal linear reassembling, with no anchor restriction, iff:

$$\beta(G, \mathcal{L}) = \min \{ \beta(G, \mathcal{L}') \mid \text{there is a vertex } w \in V \text{ and } (G, \mathcal{L}') \text{ is a linear reassembling } \beta\text{-optimal relative to anchor } w \}.$$

Similar obvious conditions apply to what it means for (G, φ) to be a β -optimal linear arrangement relative to anchor w .

Lemma 30. Let $G = (V, E)$ be a simple undirected graph and $w \in V$. Let (G, \mathcal{L}) be a linear reassembling of G anchored at w , and (G, φ) be a linear arrangement of G anchored at w , such that:

$$(G, \varphi) \text{ is induced by } (G, \mathcal{L}) \quad \text{or} \quad (G, \mathcal{L}) \text{ is induced by } (G, \varphi).$$

Conclusion: (G, \mathcal{L}) is β -optimal relative to anchor w iff (G, φ) is β -optimal relative to anchor w .

Proof. Let $d = \text{degree}(w) \geq 1$ and $\Delta = \sum \{ \text{degree}(v) \mid v \in V \text{ and } v \neq w \}$. Consider the case when arrangement (G, φ) is induced by reassembling (G, \mathcal{L}) . (We omit the case when reassembling (G, \mathcal{L}) is induced by arrangement (G, φ) , which is treated similarly.) From Definition 6,

$$\beta(G, \mathcal{L}) = d + \Delta + \sum \{ \text{degree}(X_i) \mid 1 \leq i \leq n-1 \}$$

where X_1, \dots, X_{n-1} are all the clusters of size ≥ 2 in \mathcal{L} . From Definitions 7 and 10,

$$\beta(G, \varphi) = d + \sum \{ \text{degree}(X_i) \mid 1 \leq i \leq n-1 \}.$$

Hence, both $\beta(G, \mathcal{L})$ and $\beta(G, \varphi)$ are minimized when the same quantity $\sum \{ \text{degree}(X_i) \mid 1 \leq i \leq n-1 \}$ is minimized. The desired conclusion follows. \square

⁷A similar statement applies to the α -measure: If we allowed $\text{degree}_G(w) > \text{degree}_G(w')$, then the new arrangement φ' would be such that $\alpha(G, \varphi') \leq \alpha(G, \varphi)$, but not necessarily $\alpha(G, \varphi') < \alpha(G, \varphi)$.

Remark 31. There is an obvious definition of *anchored α -optimality*, similar to that of *anchored β -optimality* above. However, results for the latter do not necessarily have counterparts for the former. In particular, the conclusion of Lemma 30 does not hold for α -optimality. Specifically, there are simple counter-examples showing the existence of a simple undirected graph $G(V, E)$ with a distinguished vertex $w \in V$ such that:

- there is a linear reassembling (G, \mathcal{L}) which is α -optimal relative to anchor w ,
- but the linear arrangement (G, φ) induced by (G, \mathcal{L}) is not α -optimal relative to anchor w .

Such a counter-example is the linear reassembling (S_7, \mathcal{B}_3) and the linear arrangement (S_7, φ_3) it induces, in Example 29, both anchored at vertex “2”: the former is α -optimal for the class of all linear reassemblings of S_7 anchored at “2”, the latter is not α -optimal for the class of all linear arrangements of S_7 anchored at “2”.

There is an examination, yet to be undertaken, of the relation between linear reassemblings (G, \mathcal{L}) and linear arrangements (G, φ) that are α -optimal relative to the same anchor, similar to our study of anchored β -optimality below. This examination we do not pursue in this report. \square

Theorem 32. *For the class of all simple undirected graphs $G = (V, E)$, each with a distinguished vertex $w \in V$, the two following problems are polynomial-time reducible to each other:*

- *the β -optimality of linear arrangements (G, φ) anchored at w ,*
- *the β -optimality of linear reassemblings (G, \mathcal{L}) anchored at w .*

More explicitly, a polynomial-time algorithm \mathcal{A} , which returns a linear reassembling (G, \mathcal{L}) [resp. a linear arrangement (G, φ)] which is β -optimal relative to anchor w can be used to return in polynomial time a linear arrangement (G, φ) [resp. a linear reassembling (G, \mathcal{L})] which is β -optimal relative to anchor w .

Proof. This is an immediate consequence of Lemma 30. \square

Definition 33 (*Auxiliary graphs*). Let $G = (V, E)$ be a simple undirected graph, with $|V| = n$ and $|E| = m$. For every $w \in V$ we define what we call an *auxiliary graph* $G_w = (V_w, E_w)$ as follows:

- $V_w := V \uplus U$ where U is a fresh set of $p = \sum \{ \text{degree}_G(v) \mid v \in V \}$ vertices.
- $E_w := E \uplus D_w$ where $D_w := \{ \overline{uw} \mid u \in U \} \cup \{ \overline{u_1 u_2} \mid u_1, u_2 \in U \text{ and } u_1 \neq u_2 \}$.

Thus, the subgraph of G_w induced by the set V is simply the original graph G , and the subgraph of G_w induced by the set $U \cup \{w\}$ is the complete graph K_{p+1} over $p+1$ vertices.

Informally, G_w is constructed from G and the complete graph K_{p+1} by identifying vertex $w \in V$ with one of the vertices of K_{p+1} . In particular, w is a cut vertex of the auxiliary graph G_w . We call w , which is the common vertex of G and K_{p+1} , the *distinguished vertex* of G_w . \square

Lemma 34. *If $G_w = (V_w, E_w)$ is the auxiliary graph for vertex $w \in V$, as constructed in Definition 33, then $|V_w| \leq n^2$ and $|E_w| \leq (n^4 - 2n^3 + 3n^2 - 2n)/2$.*

Proof. The number m of edges in G is bounded by $(n^2 - n)/2$. Hence, $p = \sum \{ \text{degree}(v) \mid v \in V \} \leq (n^2 - n)$, implying that the total number of vertices $p + n$ in G_w is $\leq (n^2 - n) + n = n^2$. The number of edges in K_p is $(p^2 - p)/2$, and in K_{p+1} it is $(p^2 + p)/2$, which is $\leq ((n^2 - n)^2 + (n^2 - n))/2 = (n^4 - 2n^3 + 2n^2 - n)/2$. Hence, the total number of edges in G_w is $m + (p^2 + p)/2 \leq (n^4 - 2n^3 + 3n^2 - 2n)/2$. \square

Let \mathcal{L} be a linear binary tree over V where, as in (A) in the opening paragraph of Section 2.3, the longest chain of nested clusters of size ≥ 2 is:

$$X_1 \subsetneq X_2 \subsetneq \dots \subsetneq X_{n-1} = V,$$

and let the corresponding singleton clusters be $\{Y_0, \dots, Y_{n-1}\}$ as determined by (B). The linear tree \mathcal{L} is uniquely determined by a sequence of vertices written in the form:

$$[v_1 \quad \dots \quad v_n]$$

where $Y_0 = \{v_1\}, Y_1 = \{v_2\}, \dots, Y_{n-1} = \{v_n\}$. We say $[v_1 \dots v_n]$ is the *vertex sequence induced* by \mathcal{L} , and \mathcal{L} the *linear reassembling* (or the *linear binary tree*) induced by the vertex sequence $[v_1 \dots v_n]$.

Similarly, if $\varphi : V \rightarrow \{1, \dots, n\}$ is a linear arrangement of V , then φ is uniquely determined by a sequence of vertices in the same form:

$$[v_1 \quad \dots \quad v_n]$$

where $\varphi^{-1}(1) = v_1, \varphi^{-1}(2) = v_2, \dots, \varphi^{-1}(n) = v_n$. We say $[v_1 \dots v_n]$ is the *vertex sequence induced* by φ , and φ the *linear arrangement induced* by the vertex sequence $[v_1 \dots v_n]$.

For the auxiliary graph G_w , whether we deal with a linear reassembling (G_w, \mathcal{L}) or a linear arrangement (G_w, φ) , it is convenient to consider sequences of the following form, which interleaves vertices and cutwidths:

$$(\diamond) \quad \mathcal{S} := [x_1 \quad (r_1, s_1) \quad x_2 \quad (r_2, s_2) \quad \dots \quad \dots \quad x_{n+p-1} \quad (r_{n+p-1}, s_{n+p-1}) \quad x_{n+p}]$$

where $\{x_1, \dots, x_{n+p}\} = V_w = \{v_1, \dots, v_n\} \cup \{u_1, \dots, u_p\}$, and for every $1 \leq i \leq n+p-1$:

$$r_i := \text{degree}_{G_w}(\{x_1, \dots, x_i\}) \quad \text{and} \quad s_i := \text{degree}_{K_{p+1}}(\{x_1, \dots, x_i\}).$$

We say the sequence \mathcal{S} in (\diamond) is the *sequence of vertices and cutwidths induced* by (G_w, \mathcal{L}) or by (G_w, φ) , whichever of the two is the case. The measure β on \mathcal{S} is:

$$\beta(\mathcal{S}) := \sum_{1 \leq i \leq n+p-1} (r_i + s_i).$$

Lemma 35. *Consider the sequence of vertices and cutwidths induced by (G_w, \mathcal{L}) or by (G_w, φ) , as just defined.*
Conclusion:

- For every $1 \leq i \leq n+p-1$, it holds that $r_i + s_i = \text{degree}_{G_w}(\{x_1, \dots, x_i\})$.
- If the sequence in (\diamond) is induced by the linear arrangement (G_w, φ) , then

$$\beta(G_w, \varphi) = \beta(\mathcal{S}) = \sum_{1 \leq i \leq n+p-1} (r_i + s_i).$$

- If the sequence in (\diamond) is induced by the linear reassembling (G_w, \mathcal{L}) , then

$$\beta(G_w, \mathcal{L}) = \Delta + \beta(\mathcal{S}) = \Delta + \sum_{1 \leq i \leq n+p-1} (r_i + s_i),$$

where $\Delta = \sum \{\text{degree}_{G_w}(v) \mid v \in V_w \text{ and } v \neq x_1\}$.

Proof. Straightforward consequence of the definitions. All details omitted. \square

We say that the sequence \mathcal{S} is *scattered* if the vertices of K_{p+1} do not occur consecutively, i.e., the vertices of K_{p+1} are interspersed with vertices of $V - \{w\}$.

Lemma 36. *Let $G_w = (V_w, E_w)$ be the auxiliary graph for vertex $w \in V$, as constructed in Definition 33. Let \mathcal{S} be the sequence of vertices and cutwidths, as in (\diamond) , induced by a β -optimal linear reassembling (G_w, \mathcal{L}) or by a β -optimal linear arrangement (G_w, φ) . **Conclusion:** \mathcal{S} is not scattered.*

In words, in a β -optimal linear reassembling (G_w, \mathcal{L}) [or in a β -optimal linear arrangement (G_w, φ) , resp.] all the vertices of K_{p+1} are reassembled consecutively [or arranged consecutively, resp.] without intervening vertices from $V - \{w\}$.

Proof. In Appendix B. □

Consider again the sequence \mathcal{S} of vertices and cutwidths in (\diamond) . Suppose \mathcal{S} is not scattered. This means that the $p + 1$ vertices of K_{p+1} occur consecutively in \mathcal{S} . We say \mathcal{S} is *balanced* iff one of two conditions holds:

$$\begin{aligned} (1) \quad & \{x_1, \dots, x_{n-1}\} = V - \{w\}, \quad \{x_n\} = \{w\}, \quad \{x_{n+1}, \dots, x_{n+p}\} = U, \\ (2) \quad & \{x_1, \dots, x_p\} = U, \quad \{x_{p+1}\} = \{w\}, \quad \{x_{p+2}, \dots, x_{n+p}\} = V - \{w\}. \end{aligned}$$

In words, \mathcal{S} is balanced if all the vertices of $V - \{w\}$ are on the same side (on the left in (1), or on the right in (2)) of the distinguished vertex w and all the vertices of U are on the other side (on the right in (1), or on the left in (2)) of w . Put differently still, \mathcal{S} is balanced if all the vertices of $V - \{w\}$ are together, all the vertices of U are together, and w is between the two sets of vertices. The following is a refinement of the preceding lemma and its proof is based on a similar argument.

Lemma 37. *Let $G_w = (V_w, E_w)$ be the auxiliary graph for vertex $w \in V$, as constructed in Definition 33. Let \mathcal{S} be the sequence of vertices and cutwidths, as in (\diamond) , induced by a β -optimal linear reassembling (G_w, \mathcal{L}) or by a β -optimal linear arrangement (G_w, φ) . **Conclusion:** \mathcal{S} is balanced.*

Proof. In Appendix B. □

By the preceding lemma, if the sequence \mathcal{S} in (\diamond) is induced by a β -optimal linear reassembling (G_w, \mathcal{L}) , or by a β -optimal linear arrangement (G_w, φ) , then \mathcal{S} is balanced, either on the left or on the right. For the rest of the analysis below, we assume that \mathcal{S} is balanced on the right, i.e., all the vertices in U occur first, then w , and then all the vertices of $V - \{w\}$.

Definition 38 (*Restrictions of linear reassemblings and linear arrangements*). Let \mathcal{L} be a linear binary tree over the set V . If $V' \subseteq V$, the *restriction* of \mathcal{L} to V' , denoted $(\mathcal{L}|V')$, consists of the following clusters:

$$(\mathcal{L}|V') := \{X \cap V' \mid X \in \mathcal{L}\}.$$

It is a straightforward exercise to show that $(\mathcal{L}|V')$ is a linear binary tree over V' .

Let $\varphi : V \rightarrow \{1, \dots, n\}$ be a linear arrangement of V . The *restriction* of φ to V' , denoted $(\varphi|V')$, is defined as follows. For every $1 \leq i \leq n' = |V'|$, let:

$$\begin{aligned} (\varphi|V')(v) &:= i \quad \text{where } v = \varphi^{-1}(j) \text{ and } j \in \{1, \dots, n\} \text{ is} \\ &\quad \text{the largest integer such that } |\{\varphi^{-1}(1), \dots, \varphi^{-1}(j-1)\} \cap V'| = i-1. \end{aligned}$$

Again here, it is straightforward to show that $(\varphi|V')$ is a linear arrangement of V' such that:

$$(\varphi|V')^{-1}(1), \dots, (\varphi|V')^{-1}(n') \quad \text{is a subsequence of} \quad \varphi^{-1}(1), \dots, \varphi^{-1}(n).$$

Moreover, if (G, \mathcal{L}) is a linear reassembling [resp. (G, φ) is a linear arrangement] of the simple undirected graph $G = (V, E)$ and $G' = (V', E')$ is the subgraph of G induced by $V' \subseteq V$, then $(G', (\mathcal{L}|V'))$ is a linear reassembling [resp. $(G', (\varphi|V'))$ is a linear arrangement] of G' . □

Lemma 39. *Let $G_w = (V_w, E_w)$ be the auxiliary graph for vertex $w \in V$, as constructed in Definition 33.*

1. *If (G_w, \mathcal{L}) is a β -optimal linear reassembling of G_w with no anchor restriction, then $(G, (\mathcal{L}|V))$ is a β -optimal linear reassembling relative to anchor w .*

2. If (G_w, φ) is a β -optimal linear arrangement of G_w with no anchor restriction, then $(G, (\varphi|V))$ is a β -optimal linear arrangement relative to anchor w .

Proof. We prove part 1 only, the proof of part 2 is similar. By Lemma 37, the sequence \mathcal{S} induced by a β -optimal linear reassembling (G_w, \mathcal{L}) is balanced. By our assumption preceding Definition 27, we take \mathcal{S} to be balanced on the right, i.e., all the vertices in U occur first, then w , and then all the vertices of $V - \{w\}$. There are no edges connecting vertices in U on the left to vertices in $V - \{w\}$ on the right, with w a cut vertex in the middle. The β -optimality of (G_w, \mathcal{L}) implies the β -optimality of the linear reassembling $(G, (\mathcal{L}|V))$ of the subgraph $G = (V, E)$ of $G_w = (V_w, E_w)$. We omit all formal details. \square

Theorem 40. *For the class of all simple undirected graphs G , the two following problems are polynomial-time reducible to each other:*

- the β -optimality of linear arrangements (G, φ) ,
- the β -optimality of linear reassemblings (G, \mathcal{L}) .

More explicitly, a polynomial-time algorithm \mathcal{A} , which returns a β -optimal linear reassembling (G, \mathcal{L}) [resp. a β -optimal linear arrangement (G, φ)] of an arbitrary graph G , can be used to return a β -optimal linear arrangement (G, φ) [resp. a β -optimal linear reassembling (G, \mathcal{L})] in polynomial time.

Proof. We compute a β -optimal linear reassembling (G_{v_i}, \mathcal{L}_i) [resp. a β -optimal linear arrangement (G_{v_i}, φ_i)] of the auxiliary graph G_{v_i} , one for each vertex $v_i \in V = \{v_1, \dots, v_n\}$. We next consider the linear reassembling $(G, (\mathcal{L}_i|V))$ [resp. the linear arrangement $(G, (\varphi_i|V))$] which, by Lemma 39, is a β -optimal linear reassembling relative to anchor v_i [resp. a β -optimal linear arrangement relative to anchor v_i], for every $1 \leq i \leq n$. Let (G, φ_i) be the linear arrangement induced by the linear reassembling $(G, (\mathcal{L}_i|V))$ [resp. let (G, \mathcal{L}_i) be the linear reassembling induced by the linear arrangement $(G, (\varphi_i|V))$]. By Lemma 30, (G, φ_i) is a β -optimal linear arrangement relative to anchor v_i [resp. (G, \mathcal{L}_i) is a β -optimal linear reassembling relative to anchor v_i], for every $1 \leq i \leq n$. Among these n linear arrangements [resp. n linear reassemblings], we choose one such that $\beta(G, \varphi_i)$ is minimized [resp. $\beta(G, \mathcal{L}_i)$ is minimized]. \square

Corollary 41. *For the class of all simple undirected graphs G , the computation of β -optimal linear reassemblings (G, \mathcal{L}) is an NP-hard problem.*

Proof. This follows from the NP-hardness of the *minimum-cost linear arrangement* problem (also called the *optimal linear arrangement* problem in the literature) [6]. This problem is the same as what we call, in this report, the problem of computing a β -optimal linear arrangement. \square

Remark 42. To the best of our knowledge, the complexity status of the *minimum-cost linear arrangement* problem (or *optimal linear arrangement* problem) for k -regular graphs for a fixed $k \geq 3$ is an open problem. If it were known to be NP-hard, we would be able to simplify our proof of Theorem 40 and its corollary considerably. \square

6 Related and Future Work

We mentioned several open problems from the literature, still unresolved to the best of our knowledge, in Remarks 25, 31, and 42. If these open problems were solved, partially or optimally, they would permit various simplifications in our proofs. In particular, even though one of our reductions can be carried out in polynomial time by invoking an earlier result on cutwidths (Lemmas 19 and 20), its $\mathcal{O}(n^{12})$ complexity is prohibitive (see the proof of Theorem 23); this is the reduction that reduces the α -optimality of linear arrangements to the α -optimality of linear reassemblings.

Beyond open problems whose resolutions would simplify and/or strengthen some of this report's results and their proofs, our wider research agenda is to tackle forms of graph reassembling other than *linear* – in particular, *balanced reassembling* and *binary reassembling* in general, both *strict* and *non-strict*, all alluded to in Sections 1, 2, and 3. For each form of reassembling, both α -optimization and β -optimization will have to be addressed; as suggested by the examination in this report, these two optimizations seem to call for different proof methods, despite their closely related definitions.

We also need to study classes of graphs for which α -optimization and/or β -optimization of their reassembling, in any of the forms mentioned above, can be carried out in low-degree polynomial times. Finally, there is the question of whether, by allowing *approximate solutions*, we can turn the NP-hardness of any of the preceding optimizations into polynomially-solvable optimizations. The literature on approximation algorithms dealing with graph layout problems is likely to be an important resource to draw from (among many other papers, the older [1, 9, 14] the more recent [4], and the survey [13]).

A Appendix: Sequential Graph Reassembling

Let \mathcal{P} be the set of all the partitions of the set $V = \{v_1, \dots, v_n\}$ of vertices in the graph $G = (V, E)$. There are two special partitions in \mathcal{P} :

$$P_0 := \{ \{v\} \mid v \in V \} \quad \text{and} \quad P_\infty := \{V\}.$$

Given two partitions $X, Y \in \mathcal{P}$, we write $X \sqsubseteq Y$ if X is *finer* than Y or, equivalently, Y is *coarser* than X , i.e., for every block $A \in X$ there is a block $B \in Y$ such that $A \subseteq B$. We write $X \sqsubset Y$ iff $X \sqsubseteq Y$ and $X \neq Y$. The relation “ \sqsubseteq ” is a (non-strict) partial order on \mathcal{P} , with a least element (the *finest* partition P_0) and a largest element (the *coarsest* partition P_∞). We need the following simple fact.

Lemma 43. *In the poset $(\mathcal{P}, \sqsubseteq)$ of all partitions of n elements, a maximal chain (linearly ordered with \sqsubseteq) is a sequence of n partitions, always starting with P_0 and ending with P_∞ .*

Proof. If $X_1 \sqsubset X_2 \sqsubset \dots \sqsubset X_k$ is a maximal chain of partitions, necessarily with $X_1 = P_0$ and $X_k = P_\infty$ because the chain is maximal, then X_1 has n blocks, X_2 has $n - 1$ blocks, in general X_p has $n - p + 1$ blocks, and X_k has one block. The length k of the chain is therefore exactly n . \square

Definition 44 (Sequential graph reassembling, i.e., according to an ordering of the edges). Let Θ be an ordering of the edges in E . We use Θ to select n partitions in \mathcal{P} forming a maximal chain (linearly ordered with \sqsubseteq), which starts with the finest partition P_0 and ends with the coarsest partition $P_\infty = \{V\}$, say:

$$X_1 \sqsubset X_2 \sqsubset \dots \sqsubset X_n \quad \text{where } X_1 = P_0 \text{ and } X_n = P_\infty,$$

as we explain next. To define X_{p+1} from X_p , we associate each X_p with a subsequence Θ_p of the initial sequence $\Theta_1 = \Theta$, for every $p \geq 1$. The subsequence Θ_p keeps track of all the edges that have not yet been reconnected. We obtain the next pair (X_{p+1}, Θ_{p+1}) from the preceding pair (X_p, Θ_p) as follows:

- (1) Take the first edge e in the sequence Θ_p , i.e., let $\Theta_p = e \Theta'_p$ for some Θ'_p , with $e = \overline{vw}$ for some $v, w \in V$, and let A and B be the unique blocks in X_p containing v and w , respectively.
- (2) Merge the two blocks A and B to obtain X_{p+1} , i.e., let:

$$X_{p+1} := (X_p - \{A, B\}) \cup \{A \cup B\}.$$

- (3) Delete every edge e' whose two endpoints are in the new block $A \cup B$ to obtain Θ_{p+1} , i.e., let:

$$\Theta_{p+1} := \Theta_p \setminus \{e' \in E \mid e' = \overline{v'w'} \text{ and } \{v', w'\} \subseteq A \cup B\}.$$

In words, we go from (X_p, Θ_p) to (X_{p+1}, Θ_{p+1}) by merging the two blocks A and B in X_p that are connected by the first edge e in Θ_p , and then removing from further consideration all edges whose endpoints are in $A \cup B$.

We refer to the sequential reassembling of G according to the ordering Θ by writing (G, Θ) , the result of which is the chain of partitions $\mathcal{X} = X_1 \sqsubset \dots \sqsubset X_n$, more succinctly written also as $\mathcal{X} = X_1 \dots X_n$. \square

Remark 45. In Definition 44, when we merge the two blocks A and B because the edge e has its two endpoints in A and B , not only do we reconnect the two halves of e , but we additionally reconnect every other edge e' whose two endpoints are also in A and B . Thus, in general, we may reconnect several edges simultaneously – all the edges between A and B in the original graph – rather than one at a time by strictly following the order specified by Θ . The same happens with binary graph-reassembling (Definition 4). \square

Definition 44 describes the process of going from an ordering Θ of edges to a maximal chain \mathcal{X} of partitions. If $\mathcal{X} = X_1 \dots X_n$ is a maximal chain of partitions, we say that \mathcal{X} is *strict* if, for every consecutive pair (X_p, X_{p+1}) with $A, B \in X_p$ and $A \cup B \in X_{p+1}$, where $1 \leq p < n$, it is the case that $\partial_G(A, B) \neq \emptyset$. The result of a sequential reassembling (G, Θ) is always a strict maximal chain \mathcal{X} of partitions. We can also carry out the process in reverse, as asserted by the next lemma.

Lemma 46 (From a maximal chain \mathcal{X} of partitions to an ordering Θ that induces it). *Let $G = (V, E)$ be a graph, and \mathcal{P} the set of all partitions of V , as in Definition 44. For every maximal chain of partitions $\mathcal{X} = X_1 \sqsubset \dots \sqsubset X_n$, with $X_1, \dots, X_n \in \mathcal{P}$, if \mathcal{X} is strict, then there is an ordering (not necessarily unique) Θ of E such that $(G, \Theta) = \mathcal{X}$.*

Proof. This is a consequence of Definition 44. Details omitted. \square

We want to relate the two notions: sequential reassembling (G, Θ) in Definition 44 and binary reassembling (G, \mathcal{B}) in Definition 4. The discussion to follow uses the following facts.

Lemma 47. *Let \mathcal{B} be a binary tree over $V = \{v_1, \dots, v_n\}$, given in the formulation of Definition 1.*

1. *If $S = \{X_1, \dots, X_k\} \subseteq \mathcal{B}$ is a maximal collection of $k \geq 2$ pairwise disjoint sets in \mathcal{B} , then S is a partition of V . We call such a maximal collection S a cross-section of the binary tree \mathcal{B} .*
2. *If \mathcal{S} is the set of all cross-sections of \mathcal{B} , then $(\mathcal{S}, \sqsubseteq)$ is a proper sub-poset of the poset $(\mathcal{P}, \sqsubseteq)$ in Lemma 43, with the same bottom element P_0 and top element P_∞ .*
3. *In the poset $(\mathcal{S}, \sqsubseteq)$, a maximal chain has exactly n entries, always starting with P_0 and ending with P_∞ .*

Proof. All three parts can be proved by induction on $n \geq 1$, using the same reasoning as in the proofs of Propositions 2 and 3. All details omitted. \square

In the preceding lemma, it is worth noting that the size of \mathcal{S} is fixed as a function of n , the so-called *Bell number* $B(n)$, which counts the partitions of an n -element set and grows exponentially in n .⁸ By contrast, the size of \mathcal{S} is much smaller, depends on both n and the shape of the binary tree \mathcal{B} , and can be as small as n (the case when \mathcal{B} is a linear, i.e., a degenerate binary tree).

Consider a sequential reassembling (G, Θ) of the graph $G = (V, E)$, the result of which is a maximal chain of n partitions $\mathcal{X} = X_1 \sqsubset \dots \sqsubset X_n$, as in Definition 44, where $X_1 = P_0$ and $X_n = P_\infty$. Since every successive partition X_{p+1} in \mathcal{X} is obtained from the previous X_p by merging two blocks in X_p , there is a natural way of organizing \mathcal{X} in the form of a binary tree \mathcal{B} , with n (not all) of the cross-sections of \mathcal{B} being exactly $\{X_1, \dots, X_n\}$. Let $\text{binary}(G, \Theta)$ denote the binary reassembling thus obtained.

Consider next a binary reassembling (G, \mathcal{B}) of the graph $G = (V, E)$. The set \mathcal{S} of all cross-sections in \mathcal{B} is uniquely defined. We want to extract from \mathcal{S} a maximal chain \mathcal{X} of cross-sections/partitions, ordered by \sqsubset , which, by Lemma 46, will in turn induce an ordering of Θ of the edges. The problem here is that there are generally many such maximal chains \mathcal{X} . We need therefore a method to canonically extract a unique maximal chain \mathcal{X} from \mathcal{S} and a unique edge-ordering Θ from \mathcal{X} . We propose such a method in the next paragraph.

We assume that the binary reassembling (G, \mathcal{B}) is strict and that there is a fixed ordering of the vertices, say, $v_1 < v_2 < \dots < v_n$. The vertex ordering “ $<$ ” is extended to edges, and to sets of edges, as follows:

- If $e = \overline{vw}$ is the edge joining vertices v and w , we assume $v < w$.
- If $e = \overline{vw}$ and $e' = \overline{v'w'}$, then $e < e'$ iff either $v < v'$ or $v = v'$ and $w < w'$.
- If $A \subseteq E$, then $\text{canon}(A)$ is the canonical ordering of A w.r.t. “ $<$ ”, i.e., $\text{canon}(A) = e_1 e_2 \dots e_k$ where $A = \{e_1, e_2, \dots, e_k\}$ and $e_1 < e_2 < \dots < e_k$.
- If A and B are non-empty disjoint set of edges, with $\text{canon}(A) = e_1 e_2 \dots$ and $\text{canon}(B) = f_1 f_2 \dots$, then $\text{canon}(A) < \text{canon}(B)$ iff $e_1 < f_1$.

If $W \in \mathcal{B}$, then \mathcal{B}_W is the subtree of \mathcal{B} rooted at W (see Proposition 3). We write (G, \mathcal{B}_W) for a *partial* binary reassembling of the graph $G = (V, E)$, the result being the subgraph of G induced by W together with all the

⁸There is no known simple expression for the exponential growth of $B(n)$ as a function of n , though there are various ways of estimating tight lower bounds and tight upper bounds on its asymptotic growth [12].

edges in $\partial_G(W)$ as dangling edges, *i.e.*, edges with only one endpoint in W . We define a canonical ordering of all the edges already in place in the partial reassembling (G, \mathcal{B}_W) , denoted $\text{canon}(G, \mathcal{B}_W)$, as follows:

$$\text{canon}(G, \mathcal{B}_W) := \begin{cases} \varepsilon \text{ (the empty string)} & \text{if } W \text{ is a singleton set,} \\ \text{canon}(G, \mathcal{B}_T) \text{ canon}(G, \mathcal{B}_U) \text{ canon}(\partial(T, U)) & \text{if } W = T \uplus U \text{ and,} \\ & \text{canon}(G, \mathcal{B}_T) < \text{canon}(G, \mathcal{B}_U), \\ \text{canon}(G, \mathcal{B}_U) \text{ canon}(G, \mathcal{B}_T) \text{ canon}(\partial(T, U)) & \text{if } W = T \uplus U \text{ and,} \\ & \text{canon}(G, \mathcal{B}_U) < \text{canon}(G, \mathcal{B}_T). \end{cases}$$

Because the binary reassembling (G, \mathcal{B}) is strict, $\partial(T, U) \neq \emptyset$ in the second and third cases above, which implies $\text{canon}(\partial(T, U)) \neq \varepsilon$. If $W = V$, then $(G, \mathcal{B}) = (G, \mathcal{B}_W)$ and $\text{canon}(G, \mathcal{B}) = \text{canon}(G, \mathcal{B}_W)$.

Proposition 48 (Relating sequential reassembling and binary reassembling). *Let $G = (V, E)$ be a simple undirected graph. We have the following facts:*

1. *For every sequential reassembling (G, Θ) , there is a binary tree \mathcal{B} over V such that:
binary(G, Θ) = (G, \mathcal{B}) .*
2. *For every strict binary reassembling (G, \mathcal{B}) , there is an ordering Θ of E such that:
canon(G, \mathcal{B}) = (G, Θ) .*
3. *For every strict binary reassembling (G, \mathcal{B}) , it holds that: binary(canon(G, \mathcal{B})) = (G, \mathcal{B}) .*

Proof. Parts 1 and 2 follow from the definitions and discussion that precede the proposition. All details omitted. Part 3 can be proved by structural induction on the subtrees \mathcal{B}_W of \mathcal{B} , where the induction hypothesis is $\text{binary}(\text{canon}(G, \mathcal{B}_W)) = (G, \mathcal{B}_W)$. All details omitted again. \square

It is possible to refine the notion of “canonical ordering” on the set of edges E , so that the equality $\text{canon}(\text{binary}(G, \Theta)) = (G, \Theta)$ holds which, together with the equality in part 3 of the preceding proposition, will mean that the functions $\text{binary}()$ and $\text{canon}()$ are inverses of each other. We omit this refinement as it will go further afield from our main concerns.

B Appendix: Remaining Proofs for Sections 4 and 5

We supply the details of several long straightforward and/or highly technical proofs which we omitted in Sections 4 and 5 in order to facilitate the grasp of the different concepts and their mutual dependence.

Proof of Theorem 26. Let $G = (V, E)$ be an arbitrary simple undirected graph, with $|V| = n$. It suffices to show that if (G, φ) is an α -optimal linear arrangement, then the linear reassembling (G, \mathcal{L}) induced by (G, φ) is also α -optimal, using Definition 11.

In the notation of Definition 11, the clusters of \mathcal{L} of size ≥ 2 are $\{X_1, \dots, X_{n-1}\}$. For the singleton clusters of \mathcal{L} , we pose $Y_{i-1} := \{\varphi^{-1}(i)\}$, where $1 \leq i \leq n$. From Definition 7:

$$\begin{aligned}\alpha(G, \varphi) &= \max \{ \text{degree}(Y_0), \max \{ \text{degree}(X_j) \mid 1 \leq j \leq n-1 \} \}, \\ \alpha(G, \mathcal{L}) &= \max \{ \max \{ \text{degree}(Y_i) \mid 0 \leq i \leq n-1 \}, \max \{ \text{degree}(X_j) \mid 1 \leq j \leq n-1 \} \}.\end{aligned}$$

By way of getting a contradiction, assume that (G, φ) is α -optimal but that the induced (G, \mathcal{L}) is not α -optimal. Hence, there is another linear reassembling (G, \mathcal{L}') which is α -optimal such that $\alpha(G, \mathcal{L}') < \alpha(G, \mathcal{L})$. Using the same notation for both (G, \mathcal{L}) and (G, \mathcal{L}') , where every name related to the latter is decorated with a prime, the inequality $\alpha(G, \mathcal{L}') < \alpha(G, \mathcal{L})$ implies the inequality:

$$\begin{aligned}&\max \{ \max \{ \text{degree}(Y'_i) \mid 0 \leq i \leq n-1 \}, \max \{ \text{degree}(X'_j) \mid 1 \leq j \leq n-1 \} \} \\ &< \max \{ \max \{ \text{degree}(Y_i) \mid 0 \leq i \leq n-1 \}, \max \{ \text{degree}(X_j) \mid 1 \leq j \leq n-1 \} \}.\end{aligned}$$

But $\max \{ \text{degree}(Y'_i) \mid 0 \leq i \leq n-1 \} = \max \{ \text{degree}(Y_i) \mid 0 \leq i \leq n-1 \}$, which implies two inequalities:

- (1) $\max \{ \text{degree}(Y_i) \mid 1 \leq i \leq n-1 \} < \max \{ \text{degree}(X_j) \mid 1 \leq j \leq n-1 \},$
- (2) $\max \{ \text{degree}(X'_j) \mid 1 \leq j \leq n-1 \} < \max \{ \text{degree}(X_j) \mid 1 \leq j \leq n-1 \}.$

Hence, by inequality (1), we have:

$$\alpha(G, \varphi) = \alpha(G, \mathcal{L}) = \max \{ \text{degree}(X_j) \mid 1 \leq j \leq n-1 \}.$$

Consider now the linear arrangement (G, φ') induced by the linear reassembling (G, \mathcal{L}') , using Definition 10. We have:

$$\alpha(G, \varphi') = \max \{ \text{degree}(Y'_0), \max \{ \text{degree}(X'_j) \mid 1 \leq j \leq n-1 \} \}.$$

If $\text{degree}(Y'_0) \geq \max \{ \text{degree}(X'_j) \mid 1 \leq j \leq n-1 \}$, then inequality (1) implies $\alpha(G, \varphi') < \alpha(G, \varphi)$, else inequality (2) implies again $\alpha(G, \varphi') < \alpha(G, \varphi)$. In both cases, the α -optimality of (G, φ) is contradicted. \square

For the proofs of Lemma 36 and Lemma 37, we take a closer look at how the vertices of K_{p+1} are positioned in the sequence \mathcal{S} in (\diamond) in Section 5. From the fact that p is the sum of all the vertex degrees in G , it follows that p is even and $p+1$ odd. From the sequence \mathcal{S} , we can extract the subsequence $(\mathcal{S} \mid K_{p+1})$ consisting of all the vertices of K_{p+1} and corresponding cutwidths:

$$(\diamond\diamond) \quad (\mathcal{S} \mid K_{p+1}) = [x_{i_1} \quad s_{i_1} \quad x_{i_2} \quad s_{i_2} \quad \cdots \quad x_{i_p} \quad s_{i_p} \quad x_{i_{p+1}}]$$

where $\{i_1, \dots, i_{p+1}\} \subseteq \{1, \dots, n+p\}$ and $\{x_{i_1}, \dots, x_{i_{p+1}}\} = \{u_1, \dots, u_p\} \cup \{w\}$. In the preceding sequence, every vertex has the same degree p in the subgraph K_{p+1} . In the full graph G_w , every vertex from K_{p+1} has again the same degree p , except for the distinguished vertex w which has degree $p+d$ where $d = \text{degree}_G(w)$. In particular, we have:

$$s_{i_1} = p, \quad s_{i_2} = 2 \cdot (p-1), \quad s_{i_3} = 3 \cdot (p-2), \quad \dots, \quad s_{i_{p-1}} = (p-1) \cdot 2, \quad s_{i_p} = p.$$

The mid-point of $(\mathcal{S} \mid K_{p+1})$ is $x_{i_{(p/2)+1}}$. The two adjacent cutwidths of the mid-point $x_{i_{(p/2)+1}}$ are:

$$s_{i_{(p/2)}} = \frac{p}{2} \cdot \left(\frac{p}{2} + 1\right) \quad \text{and} \quad s_{i_{(p/2)+1}} = \left(\frac{p}{2} + 1\right) \cdot \frac{p}{2},$$

so that also, as one can readily check:

$$s_{i_{(p/2)}} = s_{i_{(p/2)+1}} = \frac{p^2 + 2p}{4} = \max \{s_{i_1}, s_{i_2}, \dots, s_{i_p}\},$$

and the sequence of cutwidths $(s_{i_1}, \dots, s_{i_p})$ is equal to its own reverse $(s_{i_p}, \dots, s_{i_1})$. Moreover, for every j such that $1 \leq j < i_1$ or $i_{p+1} < j \leq n + p$, we have $s_j = 0$. Also, it is intuitively useful for the argument in the proof of Lemma 36 to keep in mind that:

$$\begin{aligned} (s_{i_1} - s_j) &= p, & \text{for every } 1 \leq j < i_1, \\ (s_{i_2} - s_j) &= p - 2, & \text{for every } i_1 \leq j < i_2, \\ (s_{i_3} - s_j) &= p - 4, & \text{for every } i_2 \leq j < i_3, \\ \dots & \dots & \dots \\ (s_{i_{(p/2)}} - s_j) &= 2, & \text{for every } i_{(p/2)-1} \leq j < i_{p/2}, \\ (s_{i_{(p/2)+1}} - s_j) &= 0, & \text{for every } i_{(p/2)} \leq j < i_{(p/2)+1}. \end{aligned}$$

Proof of Lemma 36. In the sequence \mathcal{S} in (\diamond) in Section 5, suppose:

- x_i is the leftmost vertex in $U \cup \{w\}$,
- x_j is the leftmost vertex in $V - \{w\}$ to the right of x_i ,
- x_ℓ is the rightmost vertex in $U \cup \{w\}$,
- x_k is the rightmost vertex in $V - \{w\}$ to the left of x_ℓ ,

where $1 \leq i \leq j \leq k \leq \ell \leq p + n$. Graphically, \mathcal{S} can be represented by:

$$\underbrace{x_1 \quad \dots \quad x_{i-1}}_{\text{all in } V - \{w\}} \quad \underbrace{x_i \quad \dots \quad x_{j-1}}_{\text{all in } U \cup \{w\}} \quad \textcircled{x_j} \quad x_{j+1} \quad \dots \quad x_{k-1} \quad \textcircled{x_k} \quad \underbrace{x_{k+1} \quad \dots \quad x_\ell}_{\text{all in } U \cup \{w\}} \quad \underbrace{x_{\ell+1} \quad \dots \quad x_{p+n}}_{\text{all in } V - \{w\}}$$

The circled vertices, x_j and x_k , are in $V - \{w\}$. If \mathcal{S} is scattered, then $1 \leq i < j$ and/or $k < \ell \leq n + p$, with the possibility that $j = k$ in which case there is only one vertex in $V - \{w\}$ inserted between all the vertices of $U \cup \{w\}$. We define:

$$\text{scatter}(\mathcal{S}) := \min \{j - i, \ell - k\} \geq 1,$$

when \mathcal{S} is scattered. If \mathcal{S} is not scattered, we set $\text{scatter}(\mathcal{S}) := 0$, so that \mathcal{S} is scattered iff $\text{scatter}(\mathcal{S}) \geq 1$. Moreover, with p even and $p + 1$ odd, it is always the case that $\text{scatter}(\mathcal{S}) \leq p/2$, so that if \mathcal{S} is scattered, then:

$$1 \leq \text{scatter}(\mathcal{S}) \leq \frac{p}{2}.$$

To complete the proof, it suffices to show that if \mathcal{S} is scattered, then we can define another sequence \mathcal{S}' from \mathcal{S} such that:

$$\beta(\mathcal{S}') < \beta(\mathcal{S}) \quad \text{and} \quad \text{scatter}(\mathcal{S}') < \text{scatter}(\mathcal{S}).$$

We obtain \mathcal{S}' from \mathcal{S} as follows:

- if $j - i \leq \ell - k$, remove x_j from the j -th position and insert it between x_{i-1} and x_i ,
- if $j - i > \ell - k$, remove x_k from the k -th position and insert it between x_ℓ and $x_{\ell+1}$.

With no loss of generality, let $j - i \leq \ell - k$. The portion of \mathcal{S} under consideration is therefore:

$$(r_{i-1}, s_{i-1}) \quad \underbrace{x_i \quad (r_i, s_i) \quad \cdots \quad (r_{j-2}, s_{j-2}) \quad x_{j-1}}_{\text{all in } U \cup \{w\}} \quad (r_{j-1}, s_{j-1}) \quad \textcircled{x_j} \quad (r_j, s_j)$$

and the order of all the vertices in the new \mathcal{S}' is:

$$\underbrace{x_1 \quad \cdots \quad x_{i-1}}_{\text{all in } V - \{w\}} \quad \textcircled{x_j} \quad \underbrace{x_i \quad \cdots \quad x_{j-1}}_{\text{all in } U \cup \{w\}} \quad x_{j+1} \quad \cdots \quad x_{k-1} \quad \textcircled{x_k} \quad \underbrace{x_{k+1} \quad \cdots \quad x_\ell}_{\text{all in } U \cup \{w\}} \quad \underbrace{x_{\ell+1} \quad \cdots \quad x_{p+n}}_{\text{all in } V - \{w\}}$$

Let $x_j = v \in V - \{w\}$ and partition $d = \text{degree}_G(v)$ into $d = d^L + d^R$, where:

- d^L is the number of vertices in $\{x_1, \dots, x_{j-1}\} \cap V$ which are connected to v ,
- d^R is the number of vertices in $\{x_{j+1}, \dots, x_{n+p}\} \cap V$ which are connected to v .

There are different cases, depending on:

- the value of $j - i$ between 1 and $p/2$,
- the value of $d^R - d^L$ between $-d$ and $+d$,
- whether the distinguished vertex w is in $\{x_i, \dots, x_{j-1}\}$ or in $\{x_{j+1}, \dots, x_\ell\}$,
- whether v is connected to w or not.

We consider only one of the cases, which is also a “worst case” to explain, and leave to the reader all the other cases, which are simple variations of this “worst case”. For the “worst case” which we choose to consider, let:

- (1) $j = i + p/2$ so that $j - i = p/2$,
- (2) $d^L = 0$ so that $d^R - d^L = d$,
- (3) w is in $\{x_{j+1}, \dots, x_\ell\}$,
- (4) there is an edge \overline{vw} connecting v and w .

With assumptions (1) to (4), as well as after:

- substituting $i + (p/2)$ for j ,
- replacing the sequence of cutwidths $s_{i-1}, s_i, \dots, s_{i+(p/2)-1}, s_{i+(p/2)}, s_{i+(p/2)+1}$ by their actual values $0, p, \dots, (p^2 + 2p - 8)/4, (p^2 + 2p)/4, (p^2 + 2p)/4$, respectively,
- and posing $r := r_{i-1}$,

the portion of \mathcal{S} under consideration becomes:

$$(r, 0) \quad \underbrace{x_i \quad (r, p) \quad \cdots \quad \left(r, (p^2 + 2p - 8)/4\right) \quad x_{i+(p/2)}}_{\text{all in } U \cup \{w\}} \quad \left(r, (p^2 + 2p)/4\right) \quad \textcircled{v} \quad \left(r + d, (p^2 + 2p)/4\right)$$

The corresponding portion in the new \mathcal{S}' is:

$$(r, 0) \quad \textcircled{v} \quad (r + d, 0) \quad \underbrace{x_i \quad (r + d, p) \quad \cdots \quad \left(r + d, (p^2 + 2p - 8)/4\right) \quad x_{i+(p/2)}}_{\text{all in } U \cup \{w\}} \quad \left(r + d, (p^2 + 2p)/4\right)$$

with all the cutwidths to the left and to the right of the shown portion being identical in \mathcal{S} and \mathcal{S}' . It is now readily seen that the value of $\beta(\mathcal{S}')$ is:

$$\beta(\mathcal{S}') = \beta(\mathcal{S}) - (p^2 + 2p)/4 + (p/2)d = \beta(\mathcal{S}) - \frac{p^2 - 2(d-1)p}{4}$$

Let $\Delta := \sum \{ \text{degree}_G(x) \mid x \in V \}$. By the construction of the auxiliary graph G_w , we have $p = \Delta$. The value of $d = \text{degree}_G(v) \leq \Delta/2$, the upper bound $\Delta/2$ being the extreme case when G is a star graph with: v at its center, $\Delta/2$ leaf vertices among $\{x_{j+1}, \dots, x_{n+p}\} \cap V$, and all other vertices of V being isolated. Hence,

$$p^2 - 2(d-1)p \leq \Delta^2 - 2\left(\frac{\Delta}{2} - 1\right)\Delta = \Delta^2 - \Delta^2 + 2\Delta = 2\Delta.$$

Hence, $\beta(\mathcal{S}') \leq \beta(\mathcal{S}) - \Delta/2$, so that $\beta(\mathcal{S}') < \beta(\mathcal{S})$ which is the desired conclusion. \square

For precision in the next proof, we introduce the measure of *unbalance*.

Definition 49 (Unbalance). Consider the sequence \mathcal{S} in (\diamond) in Section 5.

- Let $a^L \geq 0$ be the number of vertices from $V - \{w\}$ to the left of w , and $a^R \geq 0$ be the number of vertices from $V - \{w\}$ to the right of w .
- Let $b^L \geq 0$ be the number of vertices from U to the left of w , and $b^R \geq 0$ be the number of vertices from U to the right of w .

The unbalance of \mathcal{S} is measured by:

$$\text{unbal}(\mathcal{S}) := \min \{ (n - a^L - 1) + (p - b^R), (n - a^R - 1) + (p - b^L) \}.$$

The quantity $(n - a^L - 1) + (p - b^R)$ measures \mathcal{S} 's unbalance on the left, and similarly $(n - a^R - 1) + (p - b^L)$ measures \mathcal{S} 's unbalance on the right. It is useful to keep in mind that:

$$(p - b^R) + (p - b^L) = p \quad \text{and} \quad (n - a^L - 1) + (n - a^R - 1) = n - 1,$$

so that, if the quantity $(n - a^L - 1) + (p - b^R)$ or the quantity $(n - a^R - 1) + (p - b^L)$ is reduced to 0, then the other of these two quantities is increased to $n - 1 + p$. \square

Proof of Lemma 37. By Lemma 36, we can assume that \mathcal{S} is not scattered. It suffices to show that if $\text{unbal}(\mathcal{S}) \geq 1$, we can define another sequence \mathcal{S}' from \mathcal{S} such that $\beta(\mathcal{S}') < \beta(\mathcal{S})$ and $\text{unbal}(\mathcal{S}') < \text{unbal}(\mathcal{S})$. We use the notation in the proof of Lemma 36. The portion of \mathcal{S} that we examine closely is:

$$(r_{i-1}, s_{i-1}) \quad \underbrace{x_i \quad (r_i, s_i) \quad x_{i+1} \quad (r_{i+1}, s_{i+1}) \quad \cdots \quad (r_{i+p-1}, s_{i+p-1}) \quad x_{i+p}}_{\text{all in } U \cup \{w\}} \quad (r_{i+p}, s_{i+p})$$

where:

$$\begin{aligned} V - \{w\} &= \{x_1, \dots, x_{i-1}\} \cup \{x_{i+p+1}, \dots, x_{n+p}\}, & \text{with } 1 \leq i \leq n, \\ U \cup \{w\} &= \{x_i, \dots, x_{i+p}\}, & \text{with } w = x_{i+k} \text{ and } 0 \leq k \leq p. \end{aligned}$$

We partition $d = \text{degree}_G(x_{i+k}) = \text{degree}_G(w)$ into $d = d^L + d^R$, where:

- $d^L \geq 0$ is the number of vertices in $\{x_1, \dots, x_{i-1}\}$ which are connected to $w = x_{i+k}$ in G ,
- $d^R \geq 0$ is the number of vertices in $\{x_{i+p+1}, \dots, x_{n+p}\}$ which are connected to $w = x_{i+k}$ in G .

And we partition $e = \text{degree}_{K_{p+1}}(x_{i+k}) = \text{degree}_{K_{p+1}}(w)$ into $e = e^L + e^R = p$, where:

- $e^L = k$ is the number of vertices in $\{x_i, \dots, x_{i+k-1}\}$ which are connected to $w = x_{i+k}$ in K_{p+1} ,
- $e^R = (p - k)$ is the number of vertices in $\{x_{i+k+1}, \dots, x_{i+p}\}$ which are connected to $w = x_{i+k}$ in K_{p+1} .

Though not explicitly used below, it is worth noting that e^L and e^R here are the same as b^L and b^R in Definition 49 because K_{p+1} is a complete graph (but d^L and d^R are not the same as a^L and a^R). We consider 5 separate cases, $\{(a), (b), (c), (d), (e)\}$:

(a) $k = 0$, which implies $e^L = 0$ and $e^R = p$.

In case (a), because $\text{unbal}(\mathcal{S}) \neq 0$ by hypothesis, it must be that $\{x_{i+p+1}, \dots, x_{n+p}\} \neq \emptyset$. It suffices to move the vertices in $\{x_{i+p+1}, \dots, x_{n+p}\}$ to the left of $w = x_i$, also preserving their order

$$x_1, \dots, x_{i-1}, x_{i+p+1}, \dots, x_{n+p}.$$

Using a reasoning similar to that in the proof of Lemma 36, we leave it to the reader to show that $\text{unbal}(\mathcal{S}') = 0$ and $\beta(\mathcal{S}') < \beta(\mathcal{S})$ for the resulting sequence \mathcal{S}' .

(b) $k = p$, which implies $e^L = p$ and $e^R = 0$.

Case (b) is similar to case (a). Because $\text{unbal}(\mathcal{S}) \neq 0$ by hypothesis, it must be that $\{x_1, \dots, x_{i-1}\} \neq \emptyset$. In this case, we move the vertices in $\{x_1, \dots, x_{i-1}\}$ to the right of $w = x_{i+p}$. Again, we leave it to the reader to show that $\text{unbal}(\mathcal{S}') = 0$ and $\beta(\mathcal{S}') < \beta(\mathcal{S})$ for the resulting sequence \mathcal{S}' .

For the three remaining cases, we can assume that neither $k = 0$ nor $k = p$, i.e., both $w \neq x_i$ and $w \neq x_{i+p}$. Two cases of these three are:

(c) $d^L > d^R$, in which case we tranpose $w = x_{i+k}$ and x_i .

(d) $d^L < d^R$, in which case we tranpose $w = x_{i+k}$ and x_{i+p} .

By a reasoning similar to that in the proof of Lemma 36, we leave to the reader the straightforward details showing that $\beta(\mathcal{S}') < \beta(\mathcal{S})$ in both case (c) and case (d).

(e) $d^L = d^R$, in which case the value of $\beta(\mathcal{S})$ remains unchanged by tranposing $w = x_{i+k}$ and x_i , or by tranposing $w = x_{i+k}$ and x_{i+p} , and so we need an additional argument.

The additional argument for case (e), is to first tranpose $w = x_{i+k}$ and x_i , or alternatively tranpose $w = x_{i+k}$ and x_{i+p} , thus reducing case (e) to case (a), or alternatively reducing case (e) to case (b). \square

References

- [1] Sanjeev Arora, Alan Frieze, and Haim Kaplan. A New Rounding Procedure for the Assignment Problem with Applications to Dense Graph Arrangement Problems. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, pages 21–30. IEEE, 1996. 21
- [2] Azer Bestavros and Assaf Kfoury. A Domain-Specific Language for Incremental and Modular Design of Large-Scale Verifiably-Safe Flow Networks. In *Proc. of IFIP Working Conference on Domain-Specific Languages (DSL 2011)*, EPTCS Volume 66, pages 24–47, Sept 2011. 3
- [3] Daniela Bronner and Bernard Ries. An Introduction to Treewidth. Technical Report ROSE-REPORT-2006-004, Ecole Polytechnique Fédérale de Lausanne, 2006. 12
- [4] Moses Charikar, Mohammad Taghi Hajiaghayi, Howard Karloff, and Satish Rao. ℓ_2^2 Spreading Metrics for Vertex Ordering Problems. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1018–1027. Society for Industrial and Applied Mathematics, 2006. 21
- [5] Reinhard Deistel. *Graph Theory*. Springer Verlag, 2012. 12
- [6] Michael R. Garey, David S. Johnson, and Larry Stockmeyer. Some Simplified NP-Complete Graph Problems. *Theoretical Computer Science*, 1:237–267, 1976. 20
- [7] Assaf Kfoury. The Denotational, Operational, and Static Semantics of a Domain-Specific Language for the Design of Flow Networks. In *Proc. of SBLP 2011: Brazilian Symposium on Programming Languages*, Sept 2011. 3
- [8] Assaf Kfoury. The Syntax and Semantics of a Domain-Specific Language for Flow-Network Design. *Science of Computer Programming*, 93(Part A):19–38, November 2014. 3
- [9] Tom Leighton and Satish Rao. Multicommodity Max-Flow Min-Cut Theorems and their Use in Designing Approximation Algorithms. *Journal of the ACM (JACM)*, 46(6):787–832, 1999. 21
- [10] Fillia S. Makedon, Christos H. Papadimitriou, and Ivan Hal Sudborough. Topological Bandwidth. *SIAM Journal on Algebraic and Discrete Methods*, 6(3):418–444, 1985. 14
- [11] Burkhard Monien and Ivan Hal Sudborough. Min Cut is NP-Complete for Edge Weighted Trees. *Theoretical Computer Science*, 58(1–3):209–229, 1988. 14
- [12] A.M. Odlyzko. Asymptotic Enumeration Methods, 1995. 23
- [13] Jordi Petit. Addenda to the Survey of Layout Problems. *Bulletin of the EATCS*, (105):177–201, October 2011. 4, 6, 7, 21
- [14] Satish Rao and Andréa W Richa. New Approximation Techniques for Some Ordering Problems. In *SODA*, volume 98, pages 211–219, 1998. 21
- [15] Nate Soule, Azer Bestavros, Assaf Kfoury, and Andrei Lapets. Safe Compositional Equation-based Modeling of Constrained Flow Networks. In *Proc. of 4th Int’l Workshop on Equation-Based Object-Oriented Modeling Languages and Tools*, Zürich, September 2011. 3
- [16] Dimitrios M. Thilikos, Maria J. Serna, and Hans L. Bodlaender. A Polynomial Time Algorithm for the Cutwidth of Bounded Degree Graphs with Small Treewidth. In F. Meyer auf der Heide, editor, *Algorithms, ESA 2001, LNCS 2161*, pages 380–390. Springer Verlag, 2001. 13